Operating System

Windows 2000 DNS

White Paper

Abstract

This paper describes the Microsoft® Windows® 2000 operating system Domain Naming System (DNS), including design, implementation, and migration issues. It discusses new features of the Windows 2000 implementation of DNS, provides examples of DNS implementations, and describes the architectural criteria that network architects and administrators should consider when designing a DNS namespace for the Active Directory® service to provide reliable network naming services.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA

1099

CONTENTS

WHITE PAPER 1

CONTENTS..3

INTRODUCTION..5

INTRODUCTION..5

Name Services in Windows 2000..2

Name Services in Windows 2000..2

Standards and Additional Reading..2

Standards and Additional Reading..2

DNS FUNDAMENTALS..1

DNS FUNDAMENTALS..1

History of DNS..3

History of DNS..3

The Structure of DNS..4

The Structure of DNS..4

The Hierarchy of DNS: Domain Names..4

DNS and Internet..5

Resource Records..5

Distributing the Database: Zone Files and Delegation..6

Replicating the DNS database..7

Replicating the DNS database..7

Querying the Database..8

Querying the Database..8

NEW FEATURES OF THE WINDOWS 2000 DNS 8

NEW FEATURES OF THE WINDOWS 2000 DNS 8

Time to Live for Resource Records..10

Updating the DNS Database..10

Updating the DNS Database..10

Active Directory Storage and Replication Integration..11

Active Directory Storage and Replication Integration..11

The Active Directory Service Storage Model..11

The Replication Model..13

Zone Type Conversions..13

Controlling Access to Zones..13

Incremental Zone Transfer..14

Incremental Zone Transfer..14

Protocol Description..14

IXFR and DS Integration..15

Dynamic Update..15

Dynamic Update....15
Protocol Description....16
Update Algorithm....16
Dynamic Update of DNS Records 16
Secure Dynamic Update....18
Controlling Update Access to Zones and Names....21
Aging and Scavenging....22
Aging and Scavenging....22
Aging and Scavenging Parameters....23
Record Life Span....26
Scavenging Algorithm....27
Configuring Scavenging Parameters....27
Unicode Character Support....28
Unicode Character Support....28
Interoperability Considerations....28
The Domain Locator....29
The Domain Locator....29
IP/DNS Compatible Locator....31
Caching Resolver....36
Caching Resolver....36
Name Resolution....37
Name Resolution Scenarios....40
DNS Server List Management 41
Negative Caching....41
Disabling the Caching Resolver....42
Administrative Tools....42
Administrative Tools....42
DNS Manager....42
WMI Support for DNS Server Administration....42
Interoperability Issues....43
Interoperability Issues....43
Using WINS and WINSR Records....43
Using UTF-8 Characters Format....43
Receiving Non-RFC Compliant Data....44
DNS Server Performance 44
DNS Server Performance 44

DESIGNING A DNS NAMESPACE FOR THE ACTIVE DIRECTORY... 43

DESIGNING A DNS NAMESPACE FOR THE ACTIVE DIRECTORY... 43
Server Capacity Planning....45
Server Capacity Planning....45
Choosing Names....46
Choosing Names....46

Internet Access Considerations..46
Characters in Names..55
Computer Names..55
Integrating ADS with Existing DNS Structure...57
Deploying DNS to Support Active Directory..60
Deploying DNS to Support Active Directory..60
Partitioning, and Replication (Choosing your Zones)...................................60
Using Automatic Configuration...61
WINS Referral...61

SUMMARY..60

SUMMARY..60
For More Information...63
For More Information...63

GLOSSARY..61

GLOSSARY..61

DNS FUNDAMENTALS

The designers of the Microsoft ® Windows® 2000 operating system chose the Domain Name System (DNS) as the name service for the operating system. Windows 2000 Server includes an IETF standard-based Domain Name System Server. Because it is RFC compliant it is fully compatible with any other RFC compliant DNS servers. Use of the Windows 2000 Domain Name System server is not mandatory. Any DNS Server implementation supporting Service Location Resource Records (SRV RRs, as described in an Internet Draft “A DNS RR for specifying the location of services (DNS SRV)”) and Dynamic Update (RFC2136) is sufficient to provide the name service for Windows 2000–based computers[1]. However, because this implementation of DNS is designed to fully take advantage of the Windows 2000 Active Directory® service, it is the recommended DNS server for any networked organization with a significant investment in Windows or extranet partners with Windows-based systems*.* For example, while conventional DNS Servers use single-master replication, Windows 2000 DNS can be integrated into Active Directory service, so that it uses the Windows 2000 multi-master replication engine. (Note that the Active Directory supports multi-master replication.) In this way, network managers can simplify system administration by not having to maintain a separate replication topology for DNS.

DNS in Windows 2000 provides a unique DNS Server implementation that is fully interoperable with other standards-based implementations of DNS Server. Some special interoperability issues are discussed later in this paper.

The purpose of this document is to assist network architects and administrators in planning the Windows 2000 Active Directory service DNS deployment strategy. It covers the design, implementation, and migration issues that need to be considered when rolling out a scalable and robust DNS solution as a global name service.

While this paper assumes familiarity with DNS, it provides a quick overview of the DNS basics in ”DNS Fundamentals”. The Windows 2000 implementation of DNS supports various new features (as compared to Windows NT® 4.0 operating system) described in ”New Features of the Windows 2000 DNS.” It includes the description of Active Directory integration and incremental zone transfer (IXFR), dynamic (including secure) update and Unicode character support, enhanced Domain Locator, caching resolver service and DNS Manager. It provides the detailed overview of the name resolution process. It also describes the support for secure DNS management. It includes an overview of the various issues associated with designing namespace for the Active Directory. It includes integration of Active Directory with existing DNS structure and migration to the Windows 2000 implementation of DNS, design of the private namespaces and necessary DNS support.

[1] Berkeley Internet Name Domain - BIND 8.1.1 DNS Server implementation supports both SRV RRs and Dynamic Update, but it dumps core when Windows 2000-based clients send certain updates to it. 8.1.2 is the first BIND version that works reliably.

Name Services in Windows 2000

DNS is the name service of Windows 2000. It is by design a highly reliable, hierarchical, distributed, and scalable database. Windows 2000 clients use DNS for name resolution and service location, including locating domain controllers for logon.

Downlevel clients (Windows NT 3.5 and 3.51, Windows NT 4.0, Windows 95, and Windows 98), however, rely on NetBIOS which can use NBNS (WINS), broadcast or flat LmHosts file. In particular, the NetBIOS name service is used for domain controller location.

Since DNS as implemented in Windows 2000 is Windows Internet Name Services (WINS)-aware, a combination of both DNS and WINS can be used in a mixed environment to achieve maximum efficiency in locating various network services and resources. Additionally, WINS in a legacy or mixed environment plays an important interoperability role while also preserving current investment. Windows NT 4.0–based clients can register themselves in Windows 2000 WINS and Windows 2000–based clients can register in Windows NT 4.0 WINS.

Standards and Additional Reading

The following documents are of interest in the context of the Windows 2000 DNS Server implementation. They are combined in two categories. A RFC—Request For Comments—is a standard document, while Draft is work in progress that can become a standard.

RFCs:

- 1034 Domain Names—Concepts and Facilities
- 1035 Domain Names—Implementation and Specification
- 1123 Requirements for Internet Hosts—Application and Support
- 1886 DNS Extensions to Support IP Version 6
- 1995 Incremental Zone Transfer in DNS
- 1996 A Mechanism for Prompt DNS Notification of Zone Changes
- 2136 Dynamic Updates in the Domain Name System (DNS UPDATE)
- 2181 Clarifications to the DNS Specification
- 2308 Negative Caching of DNS Queries (DNS NCACHE)

Drafts:

- Draft-ietf-dnsind-rfc2052bis-02.txt (A DNS RR for Specifying the Location of Services (DNS SRV))
- Draft-skwan-utf8-dns-02.txt (Using the UTF-8 Character Set in the Domain Name System)
- Draft-ietf-dhc-dhcp-dns-08.txt (Interaction between DHCP and DNS)
- Draft-ietf-dnsind-tsig-11.txt (Secret Key Transaction Signatures for DNS (TSIG))
- Draft-ietf-dnsind-tkey-00.txt (Secret Key Establishment for DNS (TKEY RR))

- Draft-skwan-gss-tsig-04.txt (GSS Algorithm for TSIG (GSS-TSIG))

For more information on these documents, go to http://www.ietf.org/.

In addition to the listed RFCs and Drafts the implementation of the ATMA DNS records is based on the “ATM Name System Specification Version 1.0”.

Additional reading:

- Microsoft DNS and Windows NT 4.0 White Paper (http://www.microsoft.com/windows/downloads/bin/nts/DNSWP.exe)
- Designing the Active Directory Structure chapter in the Deployment Planning Guide
- Active Directory papers http://www.microsoft.com/windows2000/library/technologies/activedirectory/default.asp
- ”DNS and BIND” (Cricket Liu) published by O'Reilly and Associates, 3rd Edition ISBN: 1-56592-512-2

The Domain Name System is a hierarchical distributed database and an associated set of protocols that define:

- A mechanism for querying and updating the database
- A mechanism for replicating the information in the database among servers
- A schema of the database

History of DNS

DNS began in the early days of the Internet when the Internet was a small network established by the Department of Defense for research purposes. The host names of the computers in this network were managed through the use of a single HOSTS file located on a centrally administered server. Each site that needed to resolve host names on the network downloaded this file. As the number of hosts on the Internet grew, the traffic generated by the update process increased, as well as the size of the HOSTS file. The need for a new system, which would offer features such as scalability, decentralized administration, support for various data types, became more and more obvious.

The Domain Name System (DNS) introduced in 1984, became this new system. With DNS, the host names reside in a database that can be distributed among multiple servers, decreasing the load on any one server and providing the ability to administer this naming system on a per-partition basis. DNS supports hierarchical names and allows registration of various data types in addition to host name to IP address mapping used in HOSTS files. By virtue of the DNS database being distributed, its size is unlimited and performance does not degrade much when adding more servers.

The original DNS was based on RFC 882 (Domain names: Concepts and facilities) and RFC 883 (Domain Names–Implementation and Specification), which were superceded by RFC 1034 (Domain Names–Concepts and Facilities), and RFC 1035 (Domain Names–Implementation and Specification). RFCs that describe DNS security, implementation, and administrative issues later augmented these.

The implementation of DNS—Berkeley Internet Name Domain (BIND)—was originally developed for the 4.3 BSD UNIX operating system.

The Microsoft implementation of DNS Server became a part of the operating system in Windows NT Server 4.0. The Windows NT 4.0 DNS Server, like most DNS implementations, has its roots in RFCs 1034 and 1035.

The latest version of the Windows 2000 operating system includes a new version of DNS. The RFCs used in this version are 1034, 1035, 1886, 1996, 1995, 2136, 2308 and 2052.

The Structure of DNS

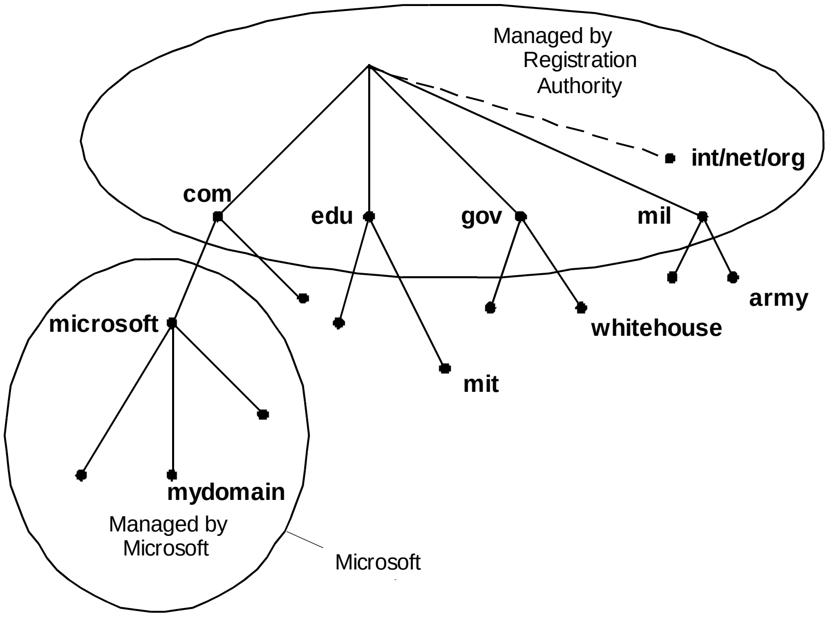
The Domain Name System is implemented as a hierarchical and distributed database containing various types of data including host names and domain names.

The names in a DNS database form a hierarchical tree structure called the *domain name space*.

The Hierarchy of DNS: Domain Names

Domain names consist of individual labels separated by dots. For example: *mydomain.microsoft.com.*

A Fully Qualified Domain Name (FQDN) uniquely identifies the host's position within the DNS hierarchical tree by specifying a list of names separated by dots on the path from the referenced host to the root. The following figure shows an example of a DNS tree with a host called *mydomain* within the *microsoft.com*. domain. The FQDN for the host would be *mydomain.microsoft.com.*



DNS and Internet

The Internet Domain Name System is managed by a Name Registration Authority on the Internet, responsible for maintaining top-level domains that are assigned by organization and by country. These domain names follow the International Standard 3166. Existing abbreviations, reserved for use by organizations, as well as two-letter and three-letter abbreviations used for countries, are shown in the following table.

DNS Domain Name	Type of Organization
com	Commercial organizations
edu	Educational institutions
org	Non-profit organizations
net	Networks (the backbone of the Internet)
gov	Non-military government organizations
mil	Military government organizations
num	Phone numbers
arpa	Reverse DNS
xx	Two-letter country code

Resource Records

A DNS database consists of resource records (RRs). Each RR identifies a particular resource within the database. There are various types of RRs in DNS.

The following table provides detailed information on structure of common RRs (Note: this is not an exhaustive list of RRs).

Description	Class	TTL	Type	Data
Start of Authority	Internet (IN)	Default TTL is 60 minutes	SOA	Owner Name, Primary Name Server DNS Name, Serial Number, Refresh Interval, Retry Interval, Expire Time, Minimum TTL
Host	Internet (IN)	Zone (SOA) TTL	A	Owner Name (Host DNS Name), Host IP Address
Name Server	Internet (IN)	Zone (SOA) TTL	NS	Owner Name, Name Server DNS Name
Mail Exchanger	Internet (IN)	Zone (SOA) TTL	MX	Owner Name, Mail Exchange Server DNS Name, Preference Number
Canonical Name (an alias)	Internet (IN)	Zone (SOA) TTL	CNAME	Owner Name (Alias Name), Host DNS Name

Distributing the Database: Zone Files and Delegation

A DNS database can be partitioned into multiple *zones.* A zone is a portion of the DNS database that contains the resource records with the owner names that belong to the contiguous portion of the DNS namespace. Zone files are maintained on DNS servers. A single DNS server can be configured to host zero, one or multiple zones.

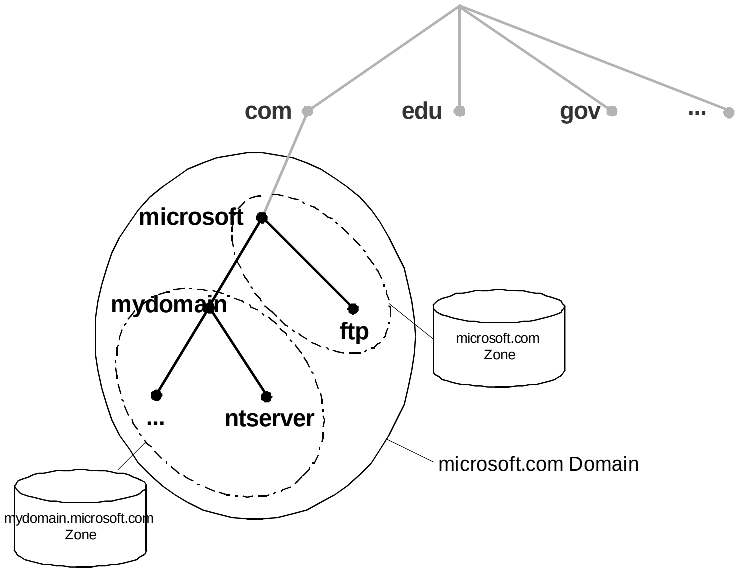
Each zone is anchored at a specific domain name referred to as the zone's *root domain*. A zone contains information about all names that end with the zone's root domain name. A DNS server is considered authoritative for a name if it loads the zone containing that name. The first record in any zone file is a Start of Authority (SOA) RR. The SOA RR identifies a primary DNS name server for the zone as the best source of information for the data within that zone and as an entity processing the updates for the zone.

Names within a zone can also be delegated to other zone(s). Delegation is a process of assigning responsibility for a portion of a DNS namespace to a separate entity. This separate entity could be another organization, department or workgroup within your company. In technical terms, delegating means assigning authority over portions of your DNS namespace to other zones. Such delegation is represented by the NS record that specifies the delegated zone and the DNS name of the server authoritative for that zone. Delegating across multiple zones was part of the original design goal of DNS. Following are the main reasons for the delegation of a DNS namespace:

- A need to delegate management of a DNS domain to a number of organizations or departments within an organization
- A need to distribute the load of maintaining one large DNS database among multiple name servers to improve the name resolution performance as well as create a DNS fault tolerant environment
- A need to allow for host's organizational affiliation by including them in appropriate domains

The NS RRs facilitate delegation by identifying DNS servers for each zone. They appear in all forward and reverse look-up zones. Whenever a DNS server needs to cross a delegation, it will refer to the NS RRs for DNS servers in the target zone.

In the figure below, the management of the *microsoft.com* domain is delegated across two zones, *microsoft.com.* and *mydomain.microsoft.com*.



Note: If multiple NS records exist for a delegated zone identifying multiple DNS servers available for querying, the Windows 2000 DNS server will be able to select the closest DNS server based on the round trip intervals measured over time for every DNS server.

Replicating the DNS database

There could be multiple zones representing the same portion of the namespace. Among these zones there are two types:

- Primary
- Secondary

Primary is a zone to which all updates for the records that belong to that zone are made. A *secondary zone* is represented by a read-only copy of the primary zone.

The changes made to the primary zone file are then replicated to the secondary zone file.

As mentioned above, a name server can host multiple zones. A server can therefore be primary for one zone (it has the master copy of the zone file) and secondary for another zone (it gets a read-only copy of the zone file).

The process of replicating a zone file to multiple name servers is *called zone transfer*. Zone transfer is achieved by copying the zone file information from the master server to the secondary server.

A *master server* is the source of the zone information. The master server can be primary or secondary. If the master is primary, then the zone transfer comes directly from the source. If the master server is secondary, the file received from the master server by means of a zone transfer is a copy of the read-only zone file.

The zone transfer is initiated in one of the following ways:

- The master server sends a *notification* (RFC 1996) to the secondary server(s) of a change in the zone.
- When the secondary server's DNS service starts or the secondary server's *refresh interval* has expired (by default it is set to 15 minutes in the SOA RR), it will query the primary server for the changes.

There are two types of zone file replication. The first, full zone transfer (AXFR), replicates the entire zone file. The second, incremental zone transfer (IXFR), replicates only the changed records of the zone. The IXFR protocol is discussed in "Incremental Zone Transfer."

BIND 4.9.3 DNS servers, as well as Windows NT 4.0 DNS, support full zone transfer (AXFR) only. There are two types of the AXFR: one requires single record per packet, the other allows multiple records per packet. The Windows 2000 DNS server supports both, but by default uses multiple records per packet, unless is configured differently for compatibility with BIND versions 4.9.4 and earlier, that do not allow multiple records per packet. The Windows 2000 DNS server supports incremental zone transfer (IXFR).

Querying the Database

DNS queries can be sent from a client (*resolver*) to a DNS server (a *name server*), or between two name servers.

A query is merely a request for records of a specified type with a specified name. For example, a query can request all host RRs with a particular name.

There are two types of queries that can be made to a DNS server:

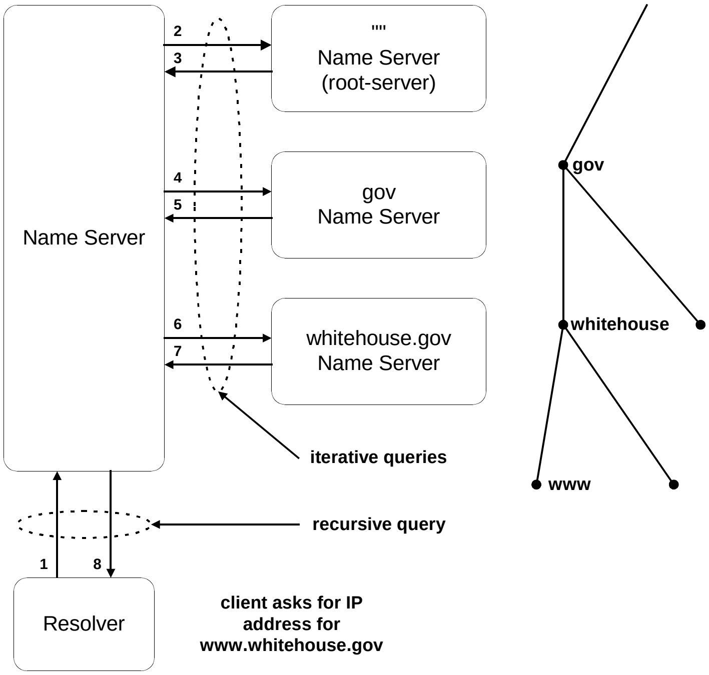
- Recursive
- Iterative

A *recursive* query forces a DNS server to respond to a request with either a failure or a successful response. Resolvers typically make recursive queries. With a recursive query, the DNS server must contact any other DNS servers it needs to resolve the request. When it receives a successful response from the other DNS Server(s), it then sends a response to the client. The recursive query is typical for a resolver querying a name server and for a name server querying its forwarder (another name server configured to handle requests forwarded to it).

When a DNS server processes a recursive query and a query can not be resolved from local zone files, the query must be escalated to a root DNS server. Each standards-based implementation of DNS includes a cache file (or root server hints) that contains entries for Root Servers of the Internet domains. The latest version of the named cache file can be downloaded from InterNIC at ftp://rs.internic.net/domain/named.cache.

An *iterative query* is one in which the name server is expected to provide the best information (also known as *referral* if the server is not authoritative for the name) based on what the server knows from local zone files or from caching. If a name server doesn't have any information to answer the query, it simply sends a negative response. A non-forwarding DNS server makes this type of query as it tries to find names outside its local domain(s). It may have to query a number of outside DNS Servers in an attempt to resolve the name.

The following figure shows an example of both types of queries.



In the provided example the following queries are used to determine IP address for www.whitehouse.gov:

- Recursive query for www.whitehouse.gov (A RR)
- Iterative query for www.whitehouse.gov (A RR)
- Referral to the gov name server (NS RRs, for gov); for simplicity iterative A queries by the DNS server (on the left) to resolve the IP addresses of the Host names of the name servers returned by other DNS servers have been omitted.
- Iterative query for www.whitehouse.gov (A RR)
- Referral to the whitehouse.gov name server (NS RR, for whitehouse.gov)
- Iterative query for www.whitehouse.gov (A RR)
- Answer from whitehouse.gov server (the IP address for www.whitehouse.gov)
- Answer from local DNS server to Resolver (the IP address for www.whitehouse.gov)

Time to Live for Resource Records

A resolver caches the information it receives when it resolves queries. These cached responses can then be used to answer subsequent queries for the same information. The cached data, however, has a limited lifetime specified in the *Time To Live* (TTL) parameter returned with the data. TTL makes sure the DNS Server doesn’t keep information for so long that it becomes out of date. TTL for the cache can be set on the DNS database (per individual RR by specifying the TTL field of the record and per zone through the minimum TTL field of the SOA record) as well as on the resolver side by specifying the maximum TTL the resolver allows to cache the resource records.

There are two competing factors to consider when setting the time to live. One is the accuracy of the cached information, the other is the DNS server’s utilization and the network traffic. If the TTL is short, then the likelihood of having old information goes down considerably, but increases the DNS servers utilization and the network traffic. If the TTL is long, the cached responses could become outdated, meaning the resolver could give false answers to queries. At the same time a long TTL decreases the DNS server’s utilization and the network traffic. If a query is answered with an entry from cache, the TTL of the entry is also passed with the response. This way the resolvers that receive the response know how long the entry is valid. The resolvers honor the TTL from the responding server; they don’t set it again based on their own TTL. Thus entries truly expire rather than live in perpetuity as they move from server to server with an updated TTL.

Updating the DNS Database

Since the RRs in the zone files are subjected to changes, they must be updated. The implementation of DNS in Windows 2000 supports both static and dynamic updates of the DNS database. The details of the dynamic update are discussed later in the paper.

The new features of Windows 2000 DNS include:

- Active Directory service Integration

- Incremental Zone Transfer (IXFR)
- Dynamic Update and Secure Dynamic Update
- Unicode Character Support
- Enhanced Domain Locator
- Enhanced Caching Resolver Service
- Enhanced DNS Manager

Active Directory Storage and Replication Integration

In addition to supporting a conventional way of maintaining and replicating DNS zone files, the implementation of DNS in Windows 2000 has the option of using the Active Directory services as the data storage and replication engine. This approach provides the following benefits:

- DNS replication will be performed by Active Directory service, so there is no need to support a separate replication topology for DNS servers.
- Active Directory service replication provides per-property replication granularity.
- Active Directory service replication is secure.
- A primary DNS server is eliminated as a single point of failure. Original DNS replication is single-master; it relies on a primary DNS server to update all the secondary servers. Unlike original DNS replication, Active Directory service replication is multi-master; an update can be made to any domain controller in it, and the change will be propagated to other domain controllers. In this way if DNS is integrated into Active Directory service the replication engine will always synchronize the DNS zone information.

Thus Active Directory service integration significantly simplifies the administration of a DNS namespace. At the same time standard zone transfer to other servers (non Windows 2000 DNS servers and previous versions of the Microsoft DNS servers) is still supported.

The Active Directory Service Storage Model

The Active Directory service is an object-oriented X.500-compliant database, which organizes resources available on your network in a hierarchical tree-like structure. This database is managed by the set of Domain Controllers (DC). The portion of the Active Directory service database for which a specific DC is authoritative is physically located on the same computer where the DC is. Every resource in Active Directory service is represented by an object. There are two distinct types of objects supported by Active Directory service:

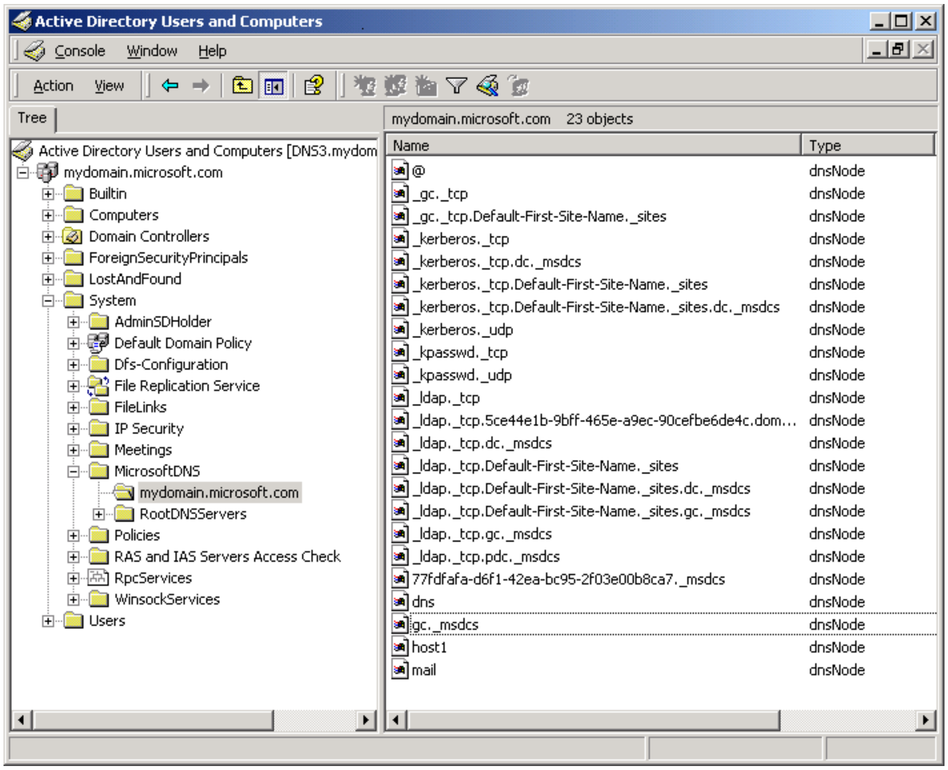
- Containers–objects that can contain other container and leaf objects
- Leafs–objects representing a specific resource within the Active Directory service tree

Each Active Directory service object has attributes associated with it that define particular characteristics of the object.

The classes of objects in the Active Directory service database as well as each object’s attributes are defined in the Active Directory service schema. In other words, the schema contains definitions for each class object available in Active Directory service. The following are examples of the Active Directory service class objects:

- User
- Group
- Organizational Unit
- DnsZone
- DnsNode

In DS integrated DNS, each DNS zone becomes an Active Directory service container object (DnsZone). The DnsZone object will contain a DnsNode leaf object for every unique name within that zone. The DnsNode object will have a DnsRecord multi-valued attribute with an instance of a value for every record associated with the object’s name.



- Active Directory Users and Computers [DNS3.mydom
  - mydomain.microsoft.com
    - Builtin
    - Computers
    - Domain Controllers
    - ForeignSecurityPrincipals
    - LostAndFound
    - System
      - AdminSDHolder
      - Default Domain Policy
      - Dfs-Configuration
      - File Replication Service
      - FileLinks
      - IP Security
      - Meetings
      - MicrosoftDNS
        - mydomain.microsoft.com
        - RootDNSServers
      - Policies
      - RAS and IAS Servers Access Check
      - RpcServices
      - WinsockServices
    - Users

mydomain.microsoft.com 23 objects

Name	Type
@	dnsNode
_gc._tcp	dnsNode
_gc._tcp.Default-First-Site-Name._sites	dnsNode
_kerberos._tcp	dnsNode
_kerberos._tcp.dc._msdcs	dnsNode
_kerberos._tcp.Default-First-Site-Name._sites	dnsNode
_kerberos._tcp.Default-First-Site-Name._sites.dc._msdcs	dnsNode
_kerberos._udp	dnsNode
_kpasswd._tcp	dnsNode
_kpasswd._udp	dnsNode
_ldap._tcp	dnsNode
_ldap._tcp.5ce44e1b-9bff-465e-a9ec-90cefbe6de4c.dom…	dnsNode
_ldap._tcp.dc._msdcs	dnsNode
_ldap._tcp.Default-First-Site-Name._sites	dnsNode
_ldap._tcp.Default-First-Site-Name._sites.dc._msdcs	dnsNode
_ldap._tcp.Default-First-Site-Name._sites.gc._msdcs	dnsNode
_ldap._tcp.gc._msdcs	dnsNode
_ldap._tcp.pdc._msdcs	dnsNode
77fdfafa-d6f1-42ea-bc95-2f03e00b8ca7._msdcs	dnsNode
dns	dnsNode
gc._msdcs	dnsNode
host1	dnsNode
mail	dnsNode

In the screen shot above, the object *mail.mydomain.microsoft.com* may have the A attribute containing the IP address for *mail.mydomain.microsoft.com.* and the MX attribute containing the mail exchange server information for *mail.mydomain.microsoft.com.*

Note: Only DNS servers running on domain controllers can load DS integrated zones.

The Replication Model

Since DNS zone information is now stored in Active Directory service, whenever an update is made to a DNS server, it simply writes the data to Active Directory and continues performing its usual functions. Active Directory service is now responsible for replicating the data to other domain controllers. The DNS servers running on other DCs will poll the updates from the DS.

Because Active Directory service uses the multi-master replication model, DNS updates can be written to any DS integrated DNS server, and the data will automatically be replicated across all the domain controllers. The multi-master replication model, however, does have some caveats that are worth discussing. The ability to write to Active Directory service from multiple domain controllers at the same time can create a conflicting situation where the changes are made to the same object on two different DNS servers. The conflict will eventually be resolved in favor of the last update made to the object based on the timestamps of the updates. The same rule is applied in the case where two or more nodes with the same name are created on two or more DNS servers. Until the conflict is resolved and the DNS server, containing invalid update, polls the valid data from the DS, it is possible that requests for the same object made to two different DNS servers will be resolved differently. This is why the ADS database is called *loosely consistent*.

Note: This subsection described the replication model between different copies of the DS integrated zones only. There are implemented two other replication models corresponding to the zone transfer between non-DS-integrated primary and secondary zone files and between DS integrated primary and secondary zone files, described below in the sections on “Protocol Description” and “IXFR and DS Integration” respectively.

Zone Type Conversions

It is possible to convert any type of existing DNS zone to any other type. The issues surrounding the primary zone conversions are of the most interest.

If a DS integrated zone is converted to an original (non-DS-integrated) primary zone file, the DNS server loading the new primary zone must become the single primary of the zone for the update. Therefore, the converted zone has to be deleted from Active Directory service (namely from all DC databases previously authoritative for this zone) so that the outdated or incorrect information is not being replicated.

Controlling Access to Zones

Active Directory service integration provides another valuable feature—the Secure Dynamic DNS Updates. The DS maintains the Access Control Lists (ACL) specifying groups or users who are allowed to modify the DS-integrated zones.

Note that only DNS server supports the Secure Dynamic Updates for the DS-integrated zones. Windows 2000 implementation provides even finer granularity allowing per-name ACL specification. More details we consider ACLs and specific Administrative groups later in “Controlling Update Access to Zones and Names.”

Incremental Zone Transfer

To reduce latency in propagation of changes to a DNS database, an algorithm has to be employed that actively notifies name servers of the change. This is accomplished by the NOTIFY extension of the DNS. The NOTIFY packet, which is sent by a Master server, does not contain any zone changes information. It merely notifies the other party that some changes have been made to a zone and that a zone transfer needs to be initiated.

The full zone transfer mechanism (AXFR) is not an efficient means to propagate changes to a zone, as it transfers the entire zone file. Incremental transfer (IXFR) is a more efficient mechanism, as it transfers only the changed portion(s) of the zone. The IXFR protocol is defined in RFC 1995.

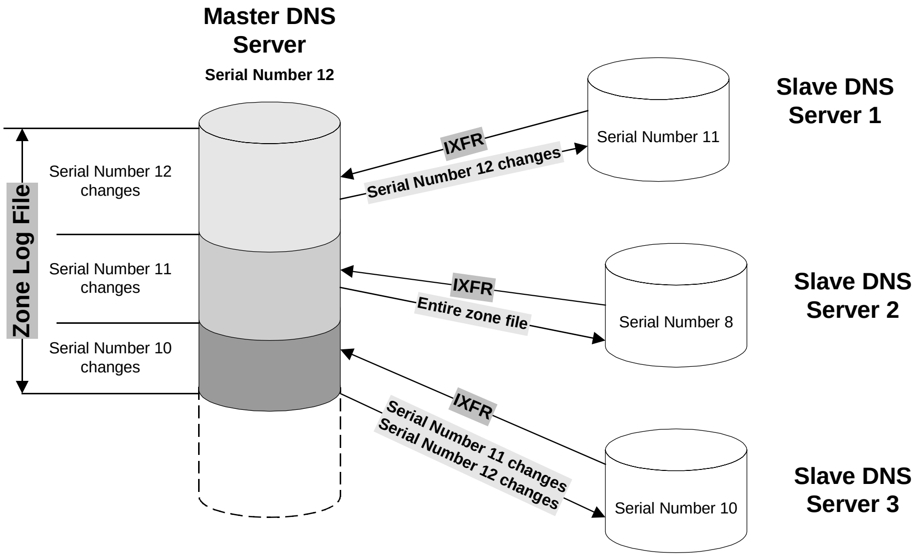
Protocol Description

When a slave name server capable of IXFR (IXFR client) initiates a zone transfer, it sends an IXFR message containing the SOA serial number of its copy of the zone.

A master name server responding to the IXFR request (IXFR server) keeps a record of the newest version of the zone and the differences between that copy and several older versions. When an IXFR request with an older serial number is received, the IXFR server sends only the changes required to make the IXFR client’s version current. In some cases, however, a full zone transfer may be chosen instead of an incremental transfer:

- The sum of the changes is larger than the entire zone.
- Only a limited number of recent changes to the zone are kept on the server for performance reasons. If the client’s serial number is lower than the one the server has in its delta changes, a full zone transfer will be initiated.
- If a name server responding to the IXFR request, does not recognize the query type, the IXFR client will automatically initiate an AXFR instead.

The following diagram details the incremental transfer mechanism.



IXFR and DS Integration

As was mentioned above, IXFR is an order-based protocol, which will send the zone changes based on differences in the zone serial numbers. In a DS integrated multi-master environment, changes to a DNS zone can be applied to any master server. Therefore, different master servers will contain the zone changes applied in a different order. This can cause problems in situations where a master IXFR server that provided the zone changes to an IXFR client the last time is not available. If the IXFR client selects another master server with zone changes applied in a different order, the integrity of the IXFR client's zone may be compromised after the incremental transfer. In this case the server initiating a zone transfer will request AXFR.

In summary, the DNS server could be a Slave and a Master with respect to the same zone at the same time. This can happen if the zone is replicated from the Master, server1, to the Slave, server2, and further from the Master, server2, to the Slave, server3. (This chain could continue further, but regardless of its length it obeys the rules described in this Section.) In this scenario the server2 will support IXFR to the server3 as long as it receives IXFR from the server1.

Dynamic Update

In a conventional DNS implementation, if the authoritative information must be changed, the network administrator has to edit the appropriate zone file manually. The Domain Name System was originally designed to support queries of a statically configured database. While the data was expected to change, the frequency of those changes was expected to be fairly low, and all updates were made as external edits to a zone's primary master file.

The advent of dynamic, automated IP addressing using DHCP and related protocols, rendered manual updating of DNS information insufficient and unusable. No human administrator can be expected to keep up with dynamic address assignments even in a medium size network environment. It was clear that automatic assignment of addresses had to be integrated with dynamic DNS updates. This capability, known as Dynamic Update, is defined in RFC 2136.

Protocol Description

The Windows 2000 DNS service supports Dynamic DNS (DDNS) as covered in RFC 2136. The RFC introduces a new *opcode* or *message format* called UPDATE. The *update message* can add and delete RRs from a specified zone as well as test for prerequisite conditions. Update is atomic, that is, all prerequisites must be satisfied or else no update operation will take place.

As in any conventional DNS implementation, the zone update must be committed on a primary name server for that zone. If an update is received by a secondary server, it will be forwarded up the replication topology until it reaches the primary server. Note that in the case of a DS integrated zone, an update for a record in that zone may be sent to any DNS server running on a domain controller whose DS contains the zone.

A zone transfer process will always lock a zone so that a secondary server gets a consistent zone view while transferring the zone data. When the zone is locked it can no longer accept dynamic updates. If the zone is large and being locked very often for the zone transfer purposes, it will starve dynamic update clients, and system can become unstable. The Windows 2000 DNS server queues the update requests that arrived during the zone transfer and processes them after the zone transfer is completed.

Update Algorithm

The update sequence consists of the following steps:

- A client, using an SOA query, locates primary DNS server and zone authoritative for the record to be registered.
- The client sends to the located DNS server an *assertion* or prerequisite-only update to verify an existing registration. If the registration does not exist, the client will send the appropriate dynamic update package to register the record.
- If the update fails the client will attempt to register the record with other primary DNS server if the authoritative zone is multimaster. If all primary DNS servers failed to process the dynamic update it will be repeated after 5 minutes and, if fails again, after another 10 minutes. If registration still failed, the described pattern of the registration attempts will be repeated after 50 minutes after the last retry.

Dynamic Update of DNS Records

Every computer running Windows 2000 attempts the registration of its A and PTR records. The service that actually generates the DNS dynamic updates is the DHCP client. The DHCP client service runs on every machine regardless of whether it is configured as DHCP client or not.

The dynamic update algorithm differs depending on the type of client network adapter engaging in the dynamic update process. The following three scenarios will be examined:

- DHCP client
- Statically configured client
- RAS client

DHCP Client

When a Windows 2000 DHCP client bootstraps, it negotiates the dynamic update procedure with a DHCP server. By default, the DHCP client always proposes that it update the A resource record, while the DHCP server updates the PTR resource record.

The Windows 2000 DHCP server can be configured to “Update DNS server according to client request” (default setting), or ”Always update forward and reverse look-ups.”

If the DHCP server is configured to Always update forward and reverse lookups, it will update both A and PTR RRs itself regardless of the DHCP client’s request.

If the DHCP server is disabled to perform dynamic updates, the DHCP client will attempt to update both A and PTR RRs itself.

At expiration of the IP address lease, these records must be removed from the appropriate zones. Dynamic cleanup requires that the records are deleted by the registering computer(s)—in this case the DHCP client or server or both—that created them. Thus, if the machine that created an A or PTR resource record is disconnected from the network before the lease expiration, the corresponding resource records may become stale. Since the DHCP server is the owner of the IP address it is encouraged that DHCP servers perform PTR records registration when possible.

Mixed Environment

It is possible that a Windows 2000 DHCP client will try to negotiate the dynamic update procedure with the Windows NT 4.0 DHCP server (or any other DHCP server that doesn’t support DNS dynamic updates). Since the Windows NT 4.0 DHCP server does not support dynamic updates, the Windows 2000 DHCP client will have to update both the A and PTR RRs itself.

In the reverse situation, with down-level clients (for example, Windows 95, Windows 98, and Windows NT 4.0), the Windows 2000 DHCP server after negotiation of a lease with a client, will register both the A and PTR records in DNS, if the ”Do updates for down-level DHCP clients” option is selected in a configuration of the DHCP server.

DHCP Server Considerations

In addition, when the DHCP client’s lease expires, the DHCP server will remove the client’s PTR RR. Also, the DHCP server will remove the corresponding A records if configured to ”Discard forward lookups when leases expire.”

Statically Configured Client

A statically configured client does not communicate with the DHCP server and dynamically updates both A and PTR RRs every time it boots up, changes its IP address or per-adapter domain name.

RAS Client

A RAS client behaves in the same manner as a statically configured client in that no interaction occurs between the client and the DHCP server. The client is responsible for dynamically updating both A and PTR RRs. The RAS client attempts to delete both records before closing the connection, but the records remain stale if the update failed for some reason (for example, the DNS server was not running at that time). The records also remain stale if the line goes down unexpectedly. In these cases a RAS server attempts deregistration of the corresponding PTR record.

Client Reregistration

One of the benefits of Dynamic Update is its ability to reregister RRs in DNS, which provides a certain level of fault tolerance in case some records in a zone become corrupted. DHCP server automatically reregisters the DNS records that it registered upon renewal of the lease. The Windows 2000-based clients reregister their DNS records every 24 hours. This value could be changed by specifying **REG_DWORD DefaultRegistrationRefreshInterval** value under the **HKLM\System\CurrentControlSet\Services\Tcpip\Parameters** registry key.

Note: When a client registers in DNS, the associated RRs include TTL, which by default is set to 20 minutes. This can be changed by specifying **REG_DWORD DefaultRegistrationTtl** value under the **HKLM\System\CurrentControlSet\Services\Tcpip\Parameters** registry key.

Dealing with Name Conflicts

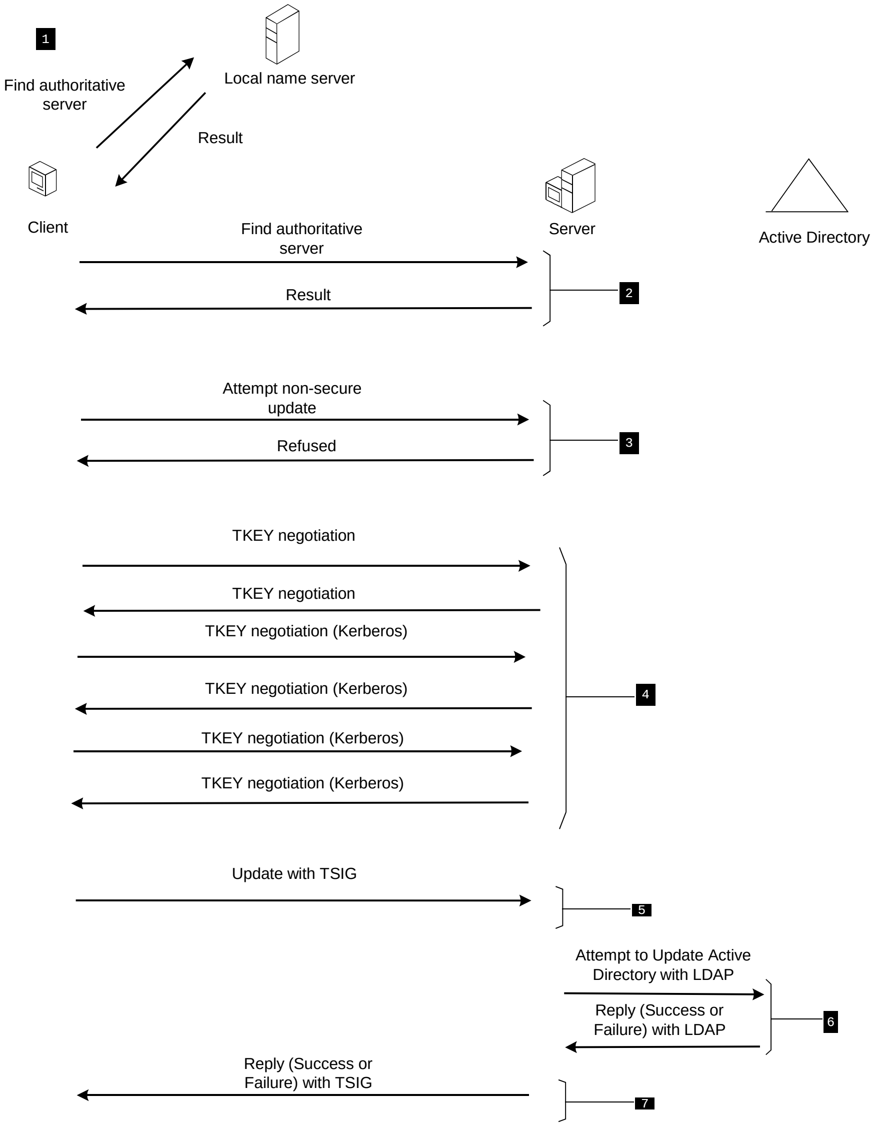
If, during Dynamic Update registration, a client discovers that its name is already registered in DNS with an IP address that belongs to some other machine, by default the client deletes the existing registration and registers its own RRs in its place. By using the appropriate registry key, this behavior may be disabled and the client will back out of the registration process and log the error in the Event Viewer. The first scenario allows you to remove stale records, but is vulnerable to malicious attacks. The second scenario has opposite effect. The problem of deletion of existing records when name collision takes place is resolved by using Secure Dynamic Updates (described in the next section). In this case only the owner of the existing record can update it.

Secure Dynamic Update

The DS integrated zones may be configured to use a Secure Dynamic Update. Access Control Lists, as mentioned in “Controlling Access to Zones,” specify the list of groups or users allowed to update resource records in such zones. The Windows 2000 DNS implementation of the Secure Dynamic Update is based on the algorithm defined in the Internet Draft "GSS Algorithm for TSIG (GSS-TSIG)." This algorithm is based on the Generic Security Service Application Program Interface (GSS-API) specified in RFC 2078. It provides security services independently of the underlying security mechanism, and separates the security services into the following processes:

- Establishing a *security context* by passing *security tokens*.
- Once a security context has been established, it has a finite lifetime during which it can be used to create and verify transaction signatures on messages between the two parties.

The sequence of events in the Secure Dynamic Update process is described below.


1
Find authoritative server
Local name server
Result
Client
Server
Active Directory
Find authoritative server
Result
2
Attempt non-secure update
Refused
3
TKEY negotiation
TKEY negotiation
TKEY negotiation (Kerberos)
TKEY negotiation (Kerberos)
TKEY negotiation (Kerberos)
TKEY negotiation (Kerberos)
4
Update with TSIG
5
Attempt to Update Active Directory with LDAP
Reply (Success or Failure) with LDAP
6
Reply (Success or Failure) with TSIG
7

In step 1, the client queries the local name server to discover which server is authoritative for the name it is attempting to update, and the local name server responds with the reference to the authoritative server.

In step 2, the client queries the authoritative server to verify that it is authoritative for the name it is attempting to update, and the server confirms it.

In step 3, the client attempts a non-secure update, and the server refuses the non-secure update. Had the server been configured for non-secure dynamic update for the appropriate zone rather than secure dynamic update, the server would have instead attempted to make the update.

In step 4, the client and server begin security context negotiation. They exchange one or more security tokens (depending on the underlying security provider) using the TKEY resource record as the vehicle to transfer security tokens between the client and the server. The TKEY resource record is specified in the Internet Draft "Secret Key Establishment for DNS (TKEY RR)."

First, the client and server negotiate an underlying security mechanism. Windows 2000 dynamic update clients and servers both propose Kerberos, so in this case, they would decide to use Kerberos. Next, using Kerberos, they verify each other's identity.

Once the security context has been established, it will be used to create and verify transaction signatures on messages between the client and server.

In step 5, the client sends the signed dynamic update request to the server. As a vehicle to transfer the signatures, the client and server use the TSIG resource record, specified in the Internet Draft "Secret Key Transaction Signatures for DNS" (TSIG).

In step 6, the server attempts to make the update to Active Directory. Whether or not it can depends on whether the client has the proper permissions to make the update and whether the prerequisites have been satisfied.

In step 7, the server sends a reply to the client stating whether or not it was able to make the update, signed with the TSIG key. If the client receives a spoofed reply, it throws it away and waits for a signed response.

Secure Dynamic Update Policy

When a client attempts a dynamic update on the DNS server, it can be configured to use one of the following approaches:

- Attempt a non-secure dynamic update first, and if it fails, negotiate a secure dynamic update (default configuration)
- Always negotiate a secure dynamic update
- Attempt only a non-secure dynamic update

The default approach is recommended as it allows client to register with a DNS servers that are not capable of the secure dynamic update. The default setting, however, can be changed through the registry.

Controlling Update Access to Zones and Names

Active Directory controls access to the secure DNS zones and names in them through the ACLs. The ACLs can be specified for either an entire zone or modified for some specific names. By default any authenticated user can create the A or PTR RRs in any zone. But once an owner name has been created (regardless of type of record) only users or groups specified in the ACL for that name with *write* permission are enabled to modify records corresponding to that name. While this approach is desirable in most scenarios, some special situations need to be considered separately.

DnsUpdateProxy Group

As described in the "Mixed Environment" section of this paper a DHCP server may be configured so that it would dynamically register A and PTR records for downlevel clients. In this situation a default configuration of the secure update may cause stale records. The following example explains. If a DHCP server performs a secure dynamic update on a name, the DHCP server becomes the owner of that name, and only that DHCP server can update the name. This can cause problems in a few circumstances. For example, suppose the DHCP server DHCP1 created an object for the name myname.mycompany.com. and then went down, and the backup DHCP server, DHCP2, tried to update the name. It would not be able to update the name because it did not own it. In a similar example, suppose DHCP1 added an object for the name myname.mycompany.com, and then the administrator upgraded the myname.mycompany.com host to Windows 2000. Because the myname.mycompany.com host did not own the name, it would not be able to update its own name.

The solution to this problem is provided by introduction of a new group called "DNS Update Proxy." Any object created by the members of this group has no security and the first user (that is not a member of the DnsUpdateProxy group) to touch a name becomes its owner. Thus, if every DHCP server registering A records for downlevel clients is a member of the DNS Update Proxy, the problem is eliminated. The DNS Update Proxy group is configurable through the Active Directory manager. At the same time, this solution introduces security holes since any DNS names registered by the computer running the DHCP server are non-secure. An A resource record for the computer is an example of such a record. The security holes may become more significant if a DHCP server (that is, a member of the DnsUpdateProxy group) is installed on a DC. In this case all SRV, A and CNAME records registered by netlogon for that DC are non-secure. To minimize the problem it is not recommended to install a DHCP server on a DC. Another strong argument against running DHCP server on a Domain Controller is, that such DHCP server has full control over all DNS objects stored in the Active Directory, since the DHCP server is running under the computer (in this case, Domain Controller) account.

DNS Admins Group

By default the DNS Admins group has full control of all zones and records in a Windows 2000 domain in which it is specified. In order for a user to be able to enumerate zones in a specific Windows 2000 domain, the user (or a group the user belongs to) must be enlisted in the DNS Admin group. At the same time it is possible that a domain administrator(s) may not want to grant such a high level of administration (full control) to all users listed in the DNS administrator group. The typical case would be if a domain administrator wanted to grant full control for a specific zone and read only control for other zones in the domain to a set of users.

Create the groups Zone1Admins, Zone2Admins, and so on for the zones 1,2, and so on respectively. Then the ACL for zone N will contain a group ZoneNAdmins with full control. At the same time all the groups Zone1Admins, Zone2Admins, and so forth will be included in the DNS Admins group. The DNS Admins group should have read permission only. Since a zone's ACL always contains the DNS Admins group, all users enlisted in the Zone1Admins, Zone2Admins, and so forth will have read permission for all the zones in the Domain.

The DNS Admins group is configurable through the Active Directory Users and Computers manager.

Reserving Names

The default configuration, where any authenticated user may create a new name in a zone, may not be sufficient for some environments requiring a high level of security. In such cases, the default ACL can be changed to allow creation of objects in a zone only by certain groups or users. *Per-name* granularity of ACLs provides another solution to this problem. An administrator may reserve a name in a zone leaving the rest of the zone open for creation of the new objects by all authenticated users. To do so an administrator needs to create a record for the reserved name and set the appropriate list of groups or users in the ACL. Then only the users listed in the ACL will be able to register another record under the reserved name.

Aging and Scavenging

With dynamic update, records are automatically added to the zone when computers and domain controllers are added. However, in some cases, they are not automatically deleted.

Having many stale resource records presents a few different problems. Stale resource records take up space on the server, and a server might use a stale resource record to answer a query. As a result, DNS server performance suffers.

To solve these problems, the Windows 2000 DNS server can *scavenge* stale records; that is, it can search the database for records that have aged and delete them. Administrators can control aging and scavenging by specifying the following:

- Which servers can scavenge zones

- Which zones can be scavenged
- Which records must be scavenged if they become stale

The DNS server uses an algorithm that ensures that it does not accidentally scavenge a record that must remain, provided that you configure all the parameters correctly. By default, the scavenging mechanism is disabled. Do not enable it unless you are absolutely certain that you understand all the parameters. Otherwise, you might accidentally configure the server to delete records that it should retain. If a name is accidentally deleted, not only do users fail to resolve queries for that name, but also, any user can create that name in DNS and then take ownership of it, even on zones configured for secure dynamic update.

You can manually enable or disable aging and scavenging on a per-server, per-zone, or per-record basis. You can also enable aging for sets of records by using Dnscmd.exe. Keep in mind that if you enable scavenging on a record that is not dynamically updated, the record will be deleted if it is not periodically refreshed, and you must recreate the record if it is still needed.

If scavenging is disabled on a standard zone and you enable scavenging, the server does not scavenge records that existed before you enabled scavenging. The server does not scavenge those records even if you convert the zone to an Active Directory–integrated zone first. To enable scavenging of such records, use Dnscmd.exe.

Aging and Scavenging Parameters

The Windows 2000 DNS server uses a record timestamp, along with parameters that you configure, to determine when to scavenge records.

The table below lists the zone parameters that affect when records are scavenged. You configure these properties on the zone.

Aging and Scavenging Parameters for Zones

Zone Parameter	Description	Configuration Tool	Notes
No-refresh interval	Time interval, after the last time a record's timestamp has been refreshed, during which the server does not accept refreshes for the record. (The server still accepts updates.)	DNS console and Dnscmd.exe	When an Active Directory–integrated zone is created, this parameter is set to the DNS server's parameter **Default no-refresh interval**. This parameter replicates through Active Directory replication.
Refresh interval	The refresh interval comes after the no-refresh interval. On expiration of the no-refresh interval, the server begins accepting refreshes for the record. After the refresh interval expires, the DNS server may scavenge records that have not been refreshed during or after the refresh interval.	DNS console and Dnscmd.exe	When an Active Directory–integrated zone is created, this parameter is set to the DNS server's parameter **DefaultRefreshInterval**. This parameter is replicated by Active Directory.
Enable Scavenging	This flag indicates whether aging and scavenging is enabled for the records in the zone.	DNS console and Dnscmd.exe	When an Active Directory–integrated zone is created, this parameter is set to the DNS server's parameter **DefaultEnableScavenging**. This parameter is replicated by Active Directory.
ScavengingServers	This parameter determines which servers can scavenge records in this zone.	Only Dnscmd.exe	This parameter is replicated by Active Directory.
Start scavenging	This parameter determines when a server can start scavenging of this zone.	Not configurable	This parameter is not replicated by Active Directory.

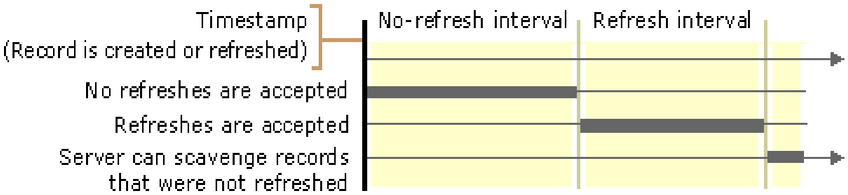
The table below lists the server parameters that affect when records are scavenged. You set these parameters on the server.

Aging and Scavenging Parameters for Servers

Server Parameter	Description	Configuration Tool	Notes
Default no-refresh interval	This value specifies the no-refresh interval that is used by default for an Active Directory–integrated zone created on this server.	DNS console (shown as **No-refresh interval**) and Dnscmd.exe	By default, this is 7 days.
Default refresh interval	This value specifies the refresh interval that is used by default for an Active Directory–integrated zone created on this server.	DNS console (shown as **Refresh interval**) and Dnscmd.exe	By default, this is 7 days.
Default Enable Scavenging	This value specifies the Enable Scavenging parameter that is used by default for an Active Directory–integrated zone created on this server.	DNS console (shown as **Enable scavenging**)and Dnscmd.exe	By default, scavenging is disabled.
Enable scavenging	This flag specifies whether the DNS server can perform scavenging of stale records. If scavenging is enabled on a server, it automatically repeats scavenging as often as specified in the **Scavenging Period** parameter.	DNS console, Advanced View (shown as **Enable automatic scavenging of stale records**) and Dnscmd.exe	By default, scavenging is disabled.
Scavenging Period	This period specifies how often a DNS server performs scavenging.	DNS console, Advanced View (shown as **Scavenging Period**) and Dnscmd.exe	By default, this is 7 days.

Record Life Span

The Figure below shows the life span of a scavengeable record.


Timestamp
(Record is created or refreshed)
No-refresh interval
Refresh interval
No refreshes are accepted
Refreshes are accepted
Server can scavenge records that were not refreshed

When a record is created or refreshed on an Active Directory–integrated zone or on a standard primary zone for which scavenging is enabled, a record’s timestamp is written.

Because of the addition of the timestamp, a standard primary zone file for which scavenging is enabled has a format slightly different from a standard DNS zone file. This does not cause any problems with zone transfer. However, you cannot copy a standard zone file for which scavenging is enabled to a non-Windows 2000-based DNS server.

The value of the timestamp is the time when the record was created or the record was last refreshed. If the record belongs to an Active Directory–integrated zone, then every time the timestamp is refreshed, the record is replicated to other domain controllers in the domain.

By default, the timestamps of records that are created by any method other than dynamic update are set to zero. A zero value indicates that the timestamp must not be refreshed and the record must not be scavenged. An Administrator can manually enable aging of such records.

After the record is refreshed, it cannot be refreshed again for the period specified by the no-refresh interval. The no-refresh interval, a zone parameter, prevents unnecessary Active Directory replication traffic.

However, the record can still be updated during the no-refresh interval. If a dynamic update request requires record modification, it is considered an update. If it does not require record modifications, it is considered a refresh. Therefore, prerequisite-only updates—updates that include a list of prerequisites but no zone changes—are also considered refreshes.

The no-refresh interval is followed by the refresh interval. After the expiration of the no-refresh interval, the server begins to accept refreshes. The record can be refreshed as long as the current time is greater than the value of the timestamp plus the no-refresh interval. When the server accepts a refresh or an update, the value of the timestamp changes to the current time.

Next, after the expiration of the refresh interval, the server can scavenge the record if it has not been refreshed. The record can be scavenged if the current time is greater than the value of the timestamp plus the value of the no-refresh interval plus the value of the refresh interval. However, the server does not necessarily scavenge the record at that time. The time at which records are scavenged depends on several server parameters.

Scavenging Algorithm

The server can be configured to perform scavenging automatically, using a fixed frequency. In addition, you can manually trigger scavenging on a server to perform immediate scavenging. When scavenging starts, the server attempts to scavenge all primary zones and succeeds if all the following conditions are met:

- The **EnableScavenging** parameter is set to **1** on the server.
- The **EnableScavenging** parameter is set to **1** on the zone.
- Dynamic update is enabled on the zone.
- The zone parameter **ScavengingServers** is not specified or contains the IP address of this server.
- The current time is greater than the value of the zone parameter **StartScavenging**.

The server sets **StartScavenging** whenever any of the following events occur:

- Dynamic update is turned on.
- **EnableScavenging** is set from **0** to **1** on the zone.
- The zone is loaded.
- The zone is resumed.

StartScavenging is equal to the time that one of the preceding events occurs plus the amount of time specified in the refresh interval for the zone. This prevents a problem that can occur if the client is unable to refresh records because the zone isn’t available—for example, if the zone is paused or the server is not working. If that happens and the server does not use **StartScavenging**, the server could scavenge the zone before the client has a chance to update the record.

When the server scavenges a zone, it examines all the records in the zone one by one. If the timestamp is not zero, and the current time is later than the time specified in the timestamp for the record plus the no-refresh and refresh intervals for the zone, it deletes the record. All other records are unaffected by the scavenging procedure.

Configuring Scavenging Parameters

This section discusses issues you must consider when configuring scavenging parameters.

To ensure that no records are deleted before the dynamic update client has time to refresh them, the refresh interval must be greater than the refresh period for each record subjected to scavenging within a zone. Many different services might refresh records at different intervals; for example, Netlogon refreshes records once an hour, cluster servers generally refresh records every 15 to 20 minutes, DHCP servers refresh records at renewal of IP address leases, and Windows 2000–based computers refresh their A and PTR resource records every 24 hours.

Usually, the DHCP service requires the longest refresh interval of all services. If you are using the Windows 2000 DHCP service, you can use the default scavenging and aging values. If you are using another DHCP server, you might need to modify the defaults.

The longer you make the no-refresh and refresh intervals, the longer stale records remain. Therefore, you might want to make those intervals as short as is reasonable. However, if you make the no-refresh interval too short, you might cause unnecessary replication by Active Directory.

Unicode Character Support

Original DNS names are restricted to the character set specified in RFCs 1123 and 952. It includes *a-z*, *0-9*, and characters. In addition, the first character of the DNS name can be a number (to accommodate the needs of companies like 3Com or 3M).

NetBIOS names are restricted to a much broader character set than the DNS names. The difference in the character sets used by the two name services could be an issue during upgrade from NetBIOS names (Windows NT 4.0) to DNS names (Windows 2000).

One solution to the problem is to rename NetBIOS names to DNS names so that they adhere to existing DNS naming standards. This is a time consuming process, which in many cases will not be possible.

The Clarification to DNS specification (RFC 2181) enlarges the character set allowed in DNS names. It specifies that a DNS label can be any binary string, and it does not necessarily have to be interpreted as ASCII. Based on this definition, Microsoft has proposed that DNS name specification be readjusted to accommodate larger character set–the UTF-8 character encoding (RFC 2044), a superset of ASCII and a translation of the UCS-2 (or Unicode) character encoding. The Windows 2000 implementation of DNS is designed to support UTF-8 character encoding.

The UTF-8 character set includes characters from most of the world’s written languages, allowing a far greater range of possible names and allowing names to use characters that are relevant to a particular locality. It solves the issue of transition from NetBIOS names (Windows NT 4.0) to DNS names (Windows 2000).

Caution is advised, however, when implementing a DNS system using the UTF-8 character encoding, as some protocols place restrictions on the characters allowed in a name. In addition, names that are intended to be globally visible (RFC 1958) should contain only the characters specified in RFC 1123.

Interoperability Considerations

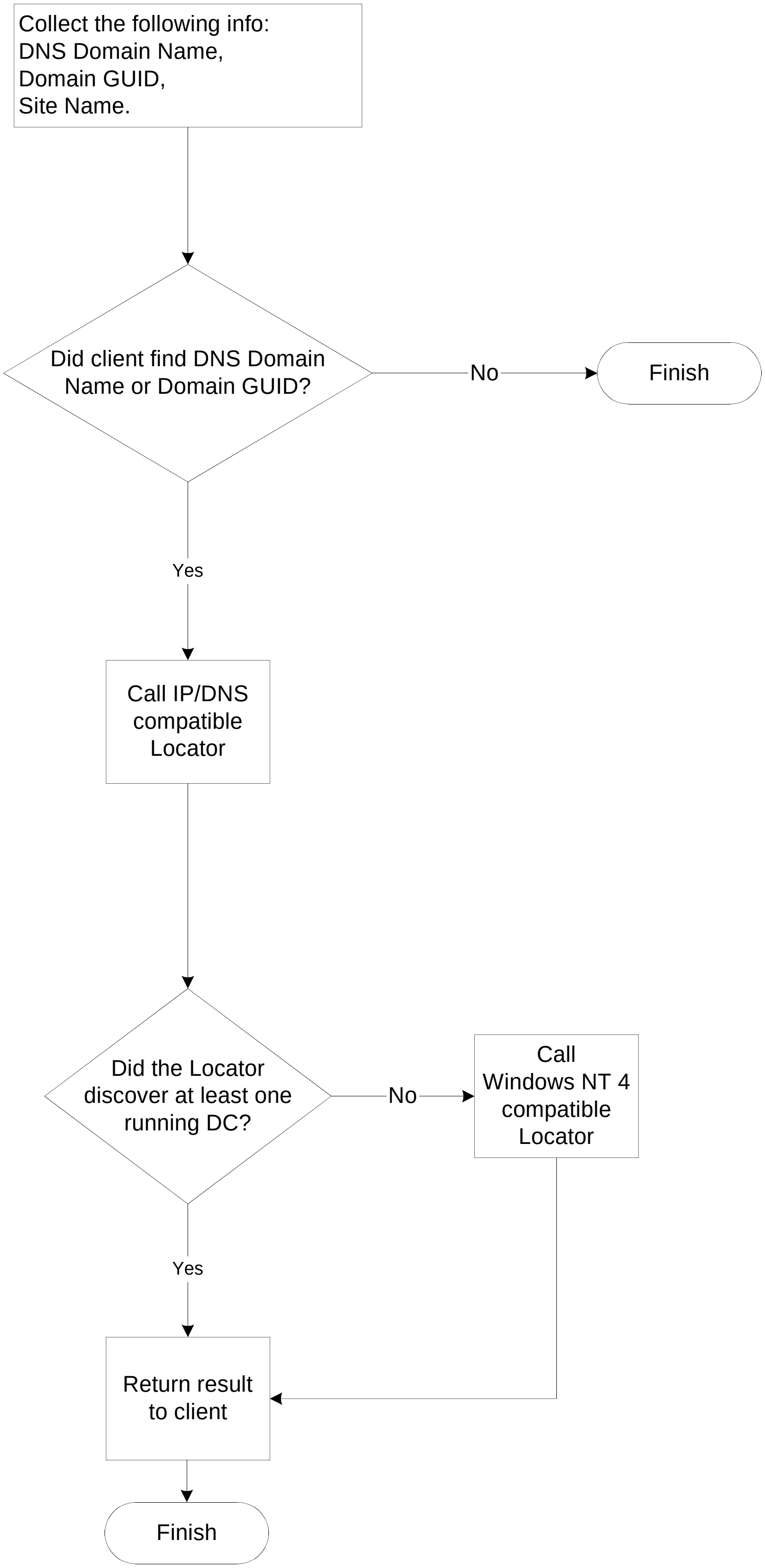
The Windows 2000 DNS server can be configured to allow or disallow the use of UTF-8 characters on a per-server or per-zone basis. A non-UTF-8–aware DNS server may accept a zone transfer of a zone containing UTF-8 names, but it may not be able to write back those names to a zone file or reload those names from a zone file. Administrators should exercise caution when transferring a zone containing UTF-8 names to a non-UTF–8-aware DNS server.

The Domain Locator

The Windows 2000 Domain Locator, implemented in the Netlogon service, is a service that enables a client (the machine locating a Domain Controller (DC)) to locate a DC. It contains the IP/DNS compatible and Windows NT 4.0 compatible locators which provide interoperability in a mixed Windows 2000- and Windows NT-based 4.0 environment.

The domain controller location algorithm, shown in the flowchart below, is implemented as follows:

- The client collects the information needed to select a domain controller:
  - The DNS domain name of the Active Directory domain the computer is joined to,
  - The domain GUID of the queried domain. It will typically only be known if the domain being queried is the primary domain of the machine. If the domain GUID is not known, it is left blank,
  - The site name. It is either obtained from a previous query or hard configuration. If neither is available, the site name is left blank.
- The NetLogon service first calls the DNS server using the IP/DNS Compatible Locator.
- If the machine running Netlogon service is not configured to use IP or DNS, or the IP/DNS Compatible Locator failed to discover a domain controller, the NetLogon service performs DC discovery using the Windows NT 4 Compatible Domain Locator.
- The information on the located domain controller is returned to the caller.


Collect the following info:
DNS Domain Name,
Domain GUID,
Site Name.
Did client find DNS Domain Name or Domain GUID?
No
Finish
Yes
Call IP/DNS compatible Locator
Did the Locator discover at least one running DC?
No
Call Windows NT 4 compatible Locator
Yes
Return result to client
Finish

The description of the Windows NT 4 Compatible Domain Locator has been omitted, since it is irrelevant to the DNS and is described in “Windows 2000 Domain Controller Locator

IP/DNS Compatible Locator

The algorithm behind the IP/DNS Compatible Locator consists of two main parts. First, the domain DC(s) must be registered with a DNS server. Second, the locator must submit a DNS query to the DNS server to locate a DC in the specified domain. After this query is resolved an LDAP User Datagram Protocol (UDP) lookup is sent to one or more of the DCs listed in the response to the DNS query to ensure their availability. Finally, the NetLogon service caches the discovered DC to aid in resolving future requests. Below this algorithm is described in detail.

DNS Record Registration and Resolver Requirements

A Windows 2000 domain is represented by a DNS domain name (for example, nt.microsoft.com.). Each domain controller registers its address with DNS using the standard DNS dynamic update. In addition to registering its host name (A record), the domain controller registers pseudonym(s) (SRV or CNAME records) that will help finding the DC based on predictable criteria (for example, the DC in a particular site). If multiple DCs have the same criteria, then there would be multiple records with the same pseudonym. A client looking for a DC with that criteria would receive all the applicable records from the DNS server.

For example, a DC named phoenix in the domain nt.microsoft.com. with an IP address of 157.55.81.157 would register the following records with DNS:

```
phoenix.nt.microsoft.com.  A          157.55.81.157
_ldap._tcp.nt.microsoft.com.          SRV     0 0 389 phoenix.nt.microsoft.com.
_kerberos._tcp.nt.microsoft.com.        SRV     0 0 88 phoenix.nt.microsoft.com.
_ldap._tcp.dc._msdcs.nt.microsoft.com.    SRV         0 0 389
phoenix.nt.microsoft.com.
_kerberos._tcp.dc._msdcs.nt.microsoft.com. SRV      0 0 88
phoenix.nt.microsoft.com.
```

With these records in place (and similar records by all the other DCs in the same domain), a simple DNS lookup of "_ldap._tcp.dc._msdcs.nt.microsoft.com." will return the names and addresses of all the DCs in the domain.

The NetLogon service on each Windows 2000 DC registers one or more of the following DNS SRV records with DNS server(s) as appropriate. The list below defines the name associated with the registered record, describes the lookup criteria supported by that record, and defines checks performed by NetLogon as each record is registered.

Netlogon registers the following DNS SRV records as appropriate:

_ldap._tcp.<DnsDomainName>.

Allows a client to find an LDAP server in the domain named by <DnsDomainName>. For example, _ldap._tcp.nt.microsoft.com. The LDAP server is not necessarily a DC. All Windows NT Domain controllers will register this name.

_ldap._tcp.<SiteName>.**_sites**.<DnsDomainName>.

Allows a client to find an LDAP server in the domain named by <DnsDomainName> and is in the site named by <SiteName>. For example, _ldap._tcp.redmond._sites.nt.microsoft.com. All Windows NT Domain controllers will register this name.

_ldap._tcp.dc._msdcs.<DnsDomainName>

Allows a client to find a DC of the domain named by <DnsDomainName>. All Windows NT Domain controllers will register this name.

_ldap._tcp.<SiteName>.**_sites.dc._msdcs**.<DnsDomainName>

Allows a client to find a DC of the domain named by <DnsDomainName> and is in the site named by <SiteName>. All Windows NT Domain controllers will register this name.

_ldap._tcp.pdc.**_msdcs**.<DnsDomainName>.

Allows a client to find the primary DC (PDC) of the domain named by <DnsDomainName>. Only the PDC of the domain registers this name. The PDC is responsible for deregistering any other registrations of this name.

_ldap._tcp.gc._msdcs.<DnsForestName>.

Allows a client to find a Global Catalog (GC) server for this domain. Only a DC serving the GC of the forest named by <DnsForestName> registers this name. For example, _ldap._tcp.gc._msdcs.microsoft.com.

_ldap._tcp.<SiteName>.**_sites.gc._msdcs**.<DnsForestName>.

Allows a client to find a Global Catalog (GC) server for this domain and is in the site named by <SiteName>. Only a DC serving the GC of the forest named by <DnsForestName> registers this name. For example, _ldap._tcp.redmond._sites.gc._msdcs.microsoft.com.

_gc._tcp.<DnsForestName>.

Allows a client to find a Global Catalog (GC) server for this domain. Only an LDAP server serving the GC of the forest named by <DnsForestName> registers this name. For example, _gc._tcp.microsoft.com. The LDAP server is not necessarily a DC.

_gc._tcp.<SiteName>**._sites.**<DnsForestName>.

Allows a client to find a Global Catalog (GC) server for this domain and is in the site named by <SiteName>. Only an LDAP server serving the GC of the forest named by <DnsForestName> registers this name. For example, _gc._tcp.redmond._sites.microsoft.com. The LDAP server is not necessarily a DC.

_ldap._tcp.<DomainGuid>.**domains._msdcs**.<DnsForestName>.

Allows a client to find a DC in a domain with a GUID of <DomainGuid>. This operation will only be done if the <DnsDomainName> of the domain has changed and the <DnsForestName> is known. This operation is expected to be infrequent. This operation will only function if the Dns Forest Name has not also been renamed. For example, _ldap._tcp.4f904480-7c78-11cf-b057-00aa006b4f8f.domains._msdcs.microsoft.com. All Windows NT Domain controllers will register this name.

_kerberos._tcp.<DnsDomainName>

Allows a client to locate a Kerberos Key Distribution Center (KDC) for the domain.

All DCs providing the Kerberos service will register this name. This service is at least an RFC-1510 compliant Kerberos 5 KDC. The KDC is not necessarily a DC. All Windows NT Domain controllers running the Kerberos KDC service will register this name.

_kerberos._udp.<DnsDomainName>

Same as **_kerberos._tcp.<DnsDomainName>** except the UDP is implied.

_kerberos._tcp.<SiteName>.**_sites.**<DnsDomainName>

Allows a client to locate a Kerberos KDC for the domain named by <DnsDomainName> and is in the site named by <SiteName>. This service is at least an RFC-1510 compliant Kerberos 5 KDC. The KDC is not necessarily a DC. All Windows NT Domain controllers running the Kerberos Key Distribution Center service will register this name.

_kerberos._tcp.dc._msdcs.<DnsDomainName>

Allows a client to find a DC running a Kerberos KDC for the domain named by <DnsDomainName>. All Windows NT Domain controllers running the Kerberos Key Distribution Center service will register this name.

_kerberos._tcp.<SiteName>.**_sites.dc._msdcs**.<DnsDomainName>

Allows a client to find a DC running a Kerberos KDC for the domain named by <DnsDomainName> and is in the site named by <SiteName>. All Windows NT Domain controllers and running the Kerberos Key Distribution Center service

_kpasswd._tcp.<DnsDomainName>

Allows a client to locate a Kerberos Password Change server for the domain. All servers providing the Kerberos Password Change service will register this name. This server at least conforms to draft-ietf-cat-kerb-chg-password-02.txt. The server is not necessarily a DC. All Windows NT Domain controllers running the Kerberos Key Distribution Center service will register this name.

_kpasswd._udp.<DnsDomainName>

Same as **_kpasswd._tcp.<DnsDomainName>** except the UDP is implied.

Netlogon registers the following DNS A records:

<DnsDomainName>.

Allows a client to find any DC in the domain via a normal A record lookup. A name such as this will be returned to the LDAP client via an LDAP referral.

gc._msdcs.<DnsForestName>

Allows a client to find any GC in the forest via a normal A record lookup. A name such as this will be returned to the LDAP client via an LDAP referral.

Netlogon registers the following DNS CNAME records:

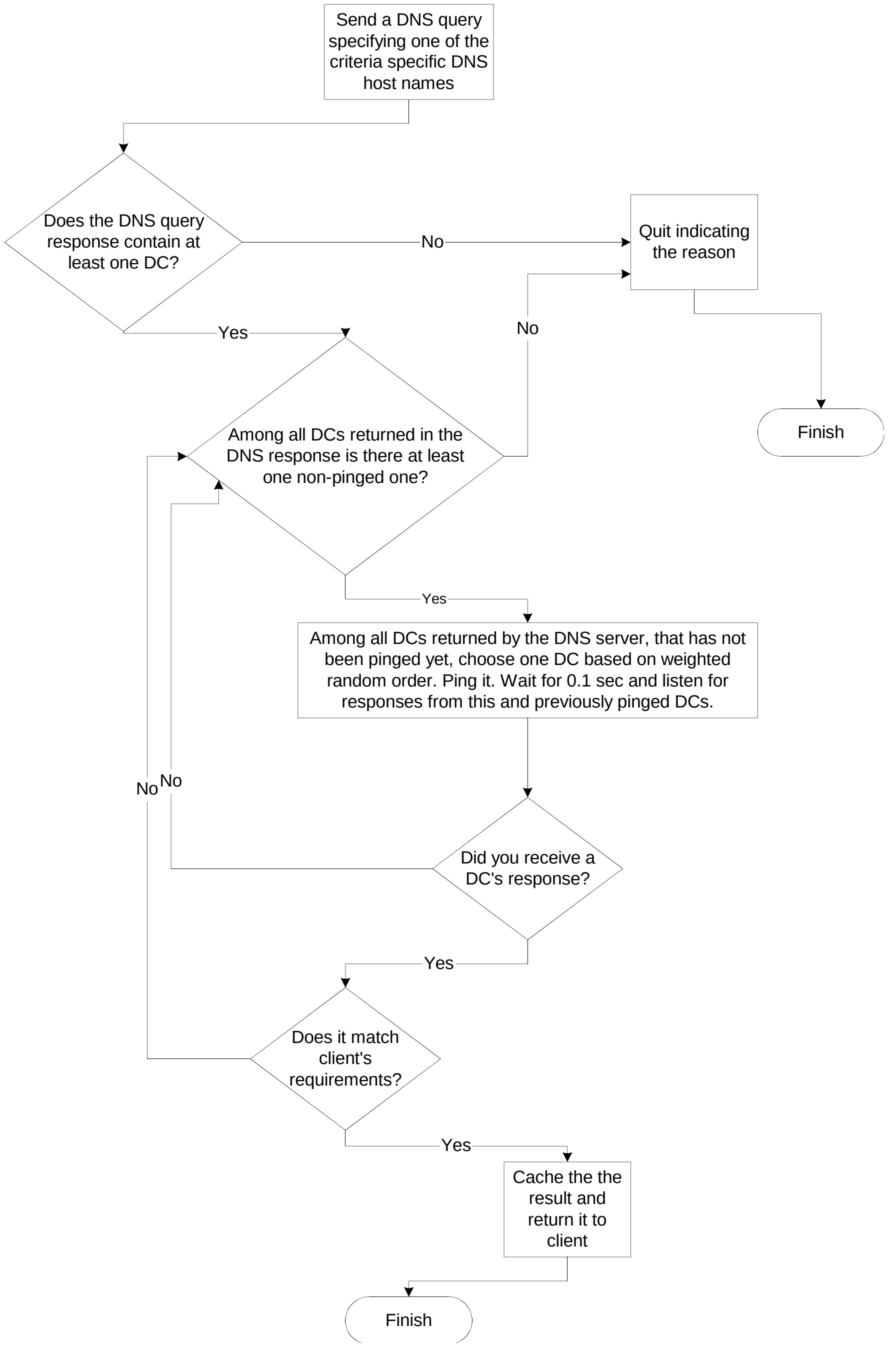
<DsaGuid>._msdcs.<DnsForestName>

Allows a client to find any DC in the forest via a normal A record lookup. The only information known about the DC is the GUID of the MSFT-DSA object for the DC and the name of the forest the DC is in. This name is used to ease the ability to rename a DC.

IP/DNS DC Locator Algorithm

The IP/DNS DC Locator algorithm is executed in the context of the NetLogon service, (typically) running on the client. The algorithm, shown in the flowchart, works as follows:

- Call DnsQuery specifying one of the criteria specific DNS host names.
- If IP is not supported or DNS is not supported, return from the algorithm indicating so.
- If the specified name cannot be found (perhaps because the domain has been renamed), return from the algorithm indicating so.
- Upon retrieving the list of DCs from DNS, the client will ping the various DCs in weighted random order. After each ping, the client will wait 1/10th second for a response to the ping. Choosing the DCs at random provides a first level of load balancing. Doing multiple pings in quick succession ensures the discovery algorithm terminates in a reasonable amount of time. The pinging continues until all the returned DCs have been tried or until positive response has been received from the pinged DC, whatever comes first.
- When a DC responds to the ping, the parameters supplied in the response is compared to the parameters required by client. If the information mismatches, the response is ignored.
- The first DC to respond to a ping and satisfy client’s requirements is used by the client.


Send a DNS query specifying one of the criteria specific DNS host names
Does the DNS query response contain at least one DC?
No
Quit indicating the reason
No
Yes
Finish
Among all DCs returned in the DNS response is there at least one non-pinged one?
Yes
Among all DCs returned by the DNS server, that has not been pinged yet, choose one DC based on weighted random order. Ping it. Wait for 0.1 sec and listen for responses from this and previously pinged DCs.
No
No
Did you receive a DC's response?
Yes
Does it match client's requirements?
Yes
Cache the the result and return it to client
Finish

Discovering Site specific DCs

When a locator searches for a DC, it attempts to find one in the same site where the client is unless specified otherwise. If at the beginning of the search the locator is not aware of the client's site, it will query a DNS server for the records of the DCs in the specified domain. Then it contacts discovered DCs and finds the site to which the client belongs. If the discovered DC is not in the same site the locator will repeat DNS query specifying the client's site.

A client might have multiple network adapters and thus might have multiple IP addresses. That could theoretically put the client in multiple sites. The design above ignores this remote possibility. Rather, it assumes that the client is in the site corresponding to the adapter, which was used to ping one of the DCs. To handle the case where this ambiguity is detrimental, NetLogon allows the site name to be manually configured by setting the REG_SZ value SiteName under the Registry key HKLM\System\CurrentControlSet\Services\NetLogon\Parameters. If this parameter is defined, the Site Name of the machine is forced to be the specified value and the site name dynamically discovered from Active Directory by the locator will never be used.

Caching Resolver

The Windows 2000 implementation of DNS introduces a client-side *caching resolver* for DNS name resolution. Caching resolver is a Windows 2000 service with the sole purpose of improving name lookup performance, and reducing network traffic associated with name lookups by minimizing the number of name resolution round trips.

Caching resolver runs in the context of the Services.exe process (Service Control Manager) and can be turned on and off just like any other service.

The Windows 2000DNS caching service includes the following features:

- General caching of queries.
- Negative caching.
- Removal of previously resolved names from the cache based on negative acknowledgement.
- Keeping track of transient (Plug and Play) network adapters and their IP configuration.
- Keeping a record of each adapter’s DomainName.
- Management of unresponsive name servers.
- The caching resolver cash can be purged using the IPCONFIG command.
- An ability to prioritize multiple A RRs returned from the DNS server based on their IP address. If the resolver sends a query specifying such prioritization, then the DNS server finds the A records with IP addresses from the networks which the computer is directly connected to and places them first in its response. This feature prevents Round Robin from working properly. It can be disabled through the Registry.
- Accepting response from non-queried IP address (This feature is enabled by default. It can be disabled through the Registry).
- An ability to automatically reload the updated local Hosts file into the resolver’s cache. Thus, as soon as a Hosts file is changed the resolver’s cache is updated.
- An ability to initiate a network failure timeout. If all resolver’s queries are timed out then it assumes the network failure and does not submit any queries for a certain period (30 seconds by default) of time. In case of a multi-adapter

computer, the same rule is applicable to every adapter separately. This feature is enabled by default. It can be disabled through the Registry.

Name Resolution

A basic name resolution request consists of a query for a given type of a DNS record with a given DNS name. The name to be resolved supplied in a query falls into one of three categories:

- Fully qualified. The name specified in the query is dot-terminated.
- Unqualified Single-Label. The name specified in the query contains no dots.
- Unqualified Multi-Label. The name specified in the query contains a dot(s), but is not dot-terminated.

Fully-Qualified Query

A fully-qualified name uniquely identifies a particular machine on the network and requires no alterations, for example *ntserver.mydomain.microsoft.com*.

If such a name needs to be resolved, first a caching resolver tries to resolve the fully-qualified query against its cache (note that the HOSTS file is preloaded in the resolver cache). If it fails then the fully-qualified query is sent directly to a DNS server. The caching resolver learns of lists of DNS servers it can query through the TCP/IP configuration of the local machine. A machine with multiple adapters may have multiple DNS server lists.

The adapters on a multi-homed machine may or may not be participating in a fully-connected network. If the networks are disjoint, the DNS namespaces on those adapters may also be disjoint. For this reason, queries must be sent to DNS servers on all adapters for complete name resolution. The response to a query can be grouped into one of four classes:

- A positive answer. The name exists and has data associated with it.
- A negative answer. The name does not exist, or the name exists, but with no associated data.
- A server failure. The server cannot service the request.
- No answer. The server does not answer within the timeout period.

The DNS servers in a list associated with a particular adapter are assumed to be members of the same namespace. Servers are queried in the order they are given in the list, which is defined by the servers priorities. If one server in the list returns a positive or negative answer, then no other servers in that list are posed the same question. The resolver may advance to the remaining servers in the list only if the current server does not respond or responds with a server failure (this scenario is slightly different for a multi-homed machine, as shown below). Should a server not respond, the resolver dynamically reorders the list changing the priority of the non-responding server (for more detailed information on this see the section on “DNS Server List Management”).

For efficiency, one fast adapter is considered the preferred adapter for name resolution. The following summarizes the name resolution algorithm:

- The query is issued to the lead server on the preferred adapter's server list.
- If no response was received within a one second interval, the query is issued to the lead server(s) on all lists, including the one on the preferred adapter.
- If no response was received within a two second interval, the query is issued to all DNS servers on all lists, including the lead servers queried before.
- If no response was received within a two second interval, once again the query is issued to all DNS servers on all lists.
- This procedure will be repeated after 4 seconds, and later after 8 seconds if no response is received.
- If the query is not resolved after all listed attempts (they may take up to seventeen seconds), then a timeout is returned to the client.

The algorithm is modified if some response(s) was received:

- If a positive response is received from a server, the response is returned to the caller and the algorithm stops
- If a negative response is received from a server, the list that server belongs to is removed from this query.
- If a server on every adapter list returns a negative response, then a negative response is returned to the caller.
- If a server returns a server failure, then that server is removed from the query for a certain period of time as described in the “DNS Server List Management” section.

Unqualified Single-Label Query

A name containing no dots is called an Unqualified Single-Label name, for example *ntserver*.

If such a name needs to be resolved it must be fully-qualified using some suffix before being placed on the wire. The list of suffixes to try can come from two places:

- Global suffix search order, and
- Primary and per-adapter domain names.

If a suffix search order is predefined, then it is used. If it is not defined then the Primary and per-adapter domain names are used.

Using Global Suffix Search Order

The global suffix search order is set by means of the TCP/IP configuration User Interface. It is not a per-adapter value. Suffixes are appended in the order given in the user interface.

The name concatenation algorithm in a name resolution process is as follows:

- The first suffix in the search order is appended to the name.

- The query is processed as a fully-qualified query.
- If the result is a positive response, the response is returned to the caller.
- If the result is a timeout, then a timeout is returned to the caller.
- If the result is a negative response, the next suffix is appended and the algorithm is restarted at step 2.
- If the suffix search list is exhausted without success, then a negative response is returned to the caller.

Using Primary and Per-adapter Domain Names

A Windows 2000-based computer has a PrimaryDnsDomainName from the machine configuration. Each adapter may also have an IpDnsDomainName from its TCP/IP configuration.

The name concatenation algorithm in a name resolution process is as follows:

1. The PrimaryDnsDomainName is appended to the name.
2. The query is submitted as a fully-qualified query

- If the result is a positive response, the response is returned to the client.
- If the result is a timeout, then a timeout is returned to the client.
- If the result is a negative response:
    - append to the original single-label name the IpDnsDomainName that has not been used yet from an adapter in the TCP/IP binding order, and the algorithm is restarted at step 2.
    - if all unique IpDnsDomainNames are exhausted, and the Registry flag for devolution is set, then the devolution algorithm is tried using the PrimaryDnsDomainName and the algorithm is restarted at step 2. Note that name devolution does not shrink primary domain name to less than a 2-label name (for example, microsoft.com.). Also note that the name devolution algorithm is not applicable to per-adapter (IpDnsDomainName) domain names.
- The response is returned to the client.

The registry key for devolution is on by default to mirror the behavior of a Windows NT 4.0-based client. Administrators may turn it off through the Registry.

Unqualified Multi-Label Query

A name containing dots, but not dot-terminated, is called an Unqualified Multi-Label name, for example *ntserver.mydomain*. A name with dots in it may be unique, or *partially* qualified.

The name resolution algorithm for such names is as follows:

- The query is submitted as a fully-qualified query (with the ntserver.mydomain. name).
- If the result is a positive response, the response is returned to the client.
- If the result is a timeout, then a timeout is returned to the client.
- If the result is a negative response, then the query is submitted as an unqualified single-label query.

- The response is returned to the client.

Name Resolution Scenarios

This section provides name resolution scenarios for a multi-homed machine using unqualified single-label and fully qualified queries. In this scenario the Global suffix search list is not specified.

The following table displays the machine’s DNS configuration:

CONFIGURATION PARAMETER		VALUE
Primary DNS Name		mydomain.microsoft.com. (default–same as domain membership)
Ethernet0 (preferred)	DNS Servers	e1, e2, e3 (from DHCP)
	DNS Domain Name	dns.microsoft.com. (from DHCP)
TokenRing0	DNS Servers	t1, t2, t3 (from DHCP)
	DNS Domain Name	dns.ntlab.microsoft.com. (from DHCP)

The Unqualified Single-Label Query Scenarios

Lookup a name on the Ethernet segment: *ping ntserver*

- Query e1 for ntserver.mydomain.microsoft.com.
- Positive response

Lookup a name on the Token Ring segment: *ping products1*

- Query e1 for products1.mydomain.microsoft.com.
- Negative response
- Query t1 for ‘products1.mydomain.microsoft.com.’
- Positive response

Lookup a name on the Corporation’s Intranet: *ping thunder*

- Query e1 for thunder.mydomain.microsoft.com.
- Negative response
- Query t1 for thunder.mydomain.microsoft.com.
- Negative response
- Query e1 for thunder.dns.microsoft.com.
- Positive response

Lookup a bogus name: *ping boguz*

- query e1 for boguz.mydomain.microsoft.com.
- negative response
- query t1 for boguz.mydomain.microsoft.com.
- negative response
- query e1 for boguz.dns.microsoft.com.

- negative response
- query t1 for boguz.dns.microsoft.com.
- negative response
- query e1 for boguz.dns.ntlab.microsoft.com.
- negative response
- query t1 for boguz.dns.ntlab.microsoft.com.
- negative response
- query e1 for boguz.microsoft.com.
- negative response
- query t1 for boguz.microsoft.com.
- negative response
- if a Registry key to send a single label query is set, then query e1 for boguz; if the response is negative, then query t1 for boguz

The Fully-Qualified Query Scenarios

Lookup a name on the Internet: *ping www.microsoft.com.*

- Query e1 for www.microsoft.com.
- Positive response

Lookup a name in the Windows NT Lab: *ping www.ntlab.microsoft.com.*

- Query e1 for www.ntlab.microsoft.com.
- Negative response
- Query t1 for www.ntlab.microsoft.com.
- Positive response

DNS Server List Management

If a DNS server does not respond to a query, its priority automatically decreases. This prevents the resolver from repeatedly timing out on servers that are not responding. However, as time goes on the priority of that DNS server could improve if it responds to further queries.

Negative Caching

Negative caching is the storage of the fact that the requested information does not exist. Just like the fact that a resource record exists and has a particular value can be cached, the fact of a non-existent resource record or name server can also be cached.

Negative caching is useful as it reduces the response time for negative answers. It also reduces the number of messages that have to be sent between resolvers and name servers, as well as network traffic generated by these messages. A large proportion of the DNS traffic on the Internet could be eliminated if all resolvers implemented negative caching.

Microsoft Implementation of Negative Caching

Microsoft’s implementation of negative caching is based on RFC 2308. It can be disabled by setting to zero the REG_DWORD NegativeCacheTime value under the

Registry key **HKEY_Local_Machine\System\CurrentControlSet\Services\DNSCache\Parameters**.

Disabling the Caching Resolver

There are two ways to disable the caching resolver:

- Manually disable the caching resolver service by typing “net stop dnscache” at the command prompt. This disables DNS server ordering, support for Plug and Play adapters, and so forth. The end result is Windows NT 4.0–like name resolution.
- Setting to zero the **REG_DWORD MaxCacheEntryTtlLimit** value that specifies maximum limit of how long the positively answered lookup is cached. This effectively eliminates caching of any RRs, but does not disable DNS server ordering and support for Plug and Play.

Administrative Tools

Windows 2000 includes various administrative tools to support DNS servers and clients. The DNS server may be administered using MMC snap-in DNS manager, command line tool dnscmd.exe and Windows Management Instrumentation (WMI).

The command line tool ipconfig.exe may be used to administer DNS client. Namely, to initiate registration of the computer A and PTR records, display or flush the cache.

DNS Manager

The Windows 2000 implementation of DNS introduces a new DNS Manager as a Microsoft Manager Console Snap-in. It provides all the functionality necessary to administer DNS server, its zones, security, and so forth.

The DNS Manager features that deserve attention are:

- The New Server Configuration Wizard, which now allows priming the root hints for a new DNS server.
- The Filtering Capability, a feature useful for the servers and zones containing a large number of zones and records, respectively.
- The new Security Capability that allows specification of the secondary servers to be notified of any changes on the master zone, as well as specification of the sets of servers to be sent the updated zone information.
- The new security capability, that allows modification of the ACLs for the DS-integrated zones and entries in such zones.

Note: In order to administer a DNS Server you need to be at least a member of the Server Operators Group on the server running DNS Server.

For more information on the DNS Manager refer to the product documentation.

WMI Support for DNS Server Administration

The Windows Management Instrumentation (WMI) provider is a set of extensions to the Windows Driver Model (WDM), an operating system interface through which hardware components can provide information and notification of events. WMI simplifies the instrumentation of various drivers and applications written for Windows, provides detailed and extensible information that is consistent across different vendors' products, and allows for consistent access to Windows instrumentation from non-Windows environments.

Among other services, WMI supports the monitoring and management of the DNS servers, zones and records. It allows enlisting and modification of the DNS servers and zones properties, enumeration of the zones and resource records, update of the resource records and creation of the new zones. The WMI allows an administrator writing an automated application managing the DNS objects. The WMI method provider enables these applications to invoke methods that are defined on the DNS server.

Interoperability Issues

In this section the issues that may arise when Microsoft DNS servers are used in the mixed environment with non-Microsoft DNS servers are discussed. Because it is RFC compliant, the Microsoft DNS server is fully interoperable with all other RFC compliant DNS servers. However, since the Microsoft DNS server provides a wider spectrum of features than specified in the RFC, the user is advised to exercise caution using these features. These features are limited to the use of WINS and WINSR resource records (as they are specified in the Windows NT 4.0 DNS white paper) and to the use of the UTF-8 character encoding.

Using WINS and WINSR Records

Since currently only Microsoft DNS servers support the WINS and WINSR resource records we recommend disabling replication of these records if all following conditions are satisfied:

- the primary copy of the zone contains one of these records;
- at least one of the secondaries resides on the non-Microsoft DNS server.

At the same time, if the secondaries reside partially on Microsoft and non-Microsoft DNS servers, disabling WINS and WINSR resource records replication may require manual input of these records to the secondary zones residing on the Microsoft DNS servers.

Using UTF-8 Characters Format

The Windows 2000 DNS server can be configured to allow or disallow the use of UTF-8 characters on a per-server or per-zone basis. A non-UTF-8-aware DNS server may accept a zone transfer of a zone containing UTF-8 names, but it may not be able to write back those names to a zone file or reload those names from a zone file. Administrators should exercise caution when transferring a zone containing UTF-8 names to a non-UTF-8-aware DNS server.

Receiving Non-RFC Compliant Data

If a Windows 2000 server supports a secondary zone and receives unknown resource records, then it drops such records and continues zone replication. It also drops a circular CNAME resource records if receives them.

DNS Server Performance

The statistics presented below are compiled as a profile of DNS server performance during preliminary testing of Windows 2000 Server. In testing, two different DNS server hardware configurations were used and overall DNS query and dynamic update activity was measured, along with processor utilization.

The results of each of these tests are listed in the table below.

Server configuration	Queries/sec	Dynamic updates/sec	Processor utilization
Intel P-II 400 MHz dual-processor	900	100	30%

During these measurements, the monitored DNS server was processing both queries and dynamic updates at the same time. The numbers above reflect these concurrent processes. The statistics were collected over a period of a few hours.

For dynamic updates, standard primary type zones were used, not Active Directory-integrated zones. Where directory integration is used for zones, the rate of the dynamic updates that can be processed decreases by a factor of 2, since the DNS server must write to the Active Directory database as well.

In addition, if a zone is configured to accept only secure dynamic updates, the update rate decreases by 25% compared to the rate of updates processed by the DNS server for the Active Directory-integrated zones that allow non-secure dynamic updates. Network performance might also be a factor in these cases since the directory database might require network activity to process updates.

The previous measurements are not meant in any way to indicate maximum performance or server limitations for Windows 2000 DNS servers. The objective of the tests was merely to sample typical DNS server performance and obtain a working benchmark based on standard available hardware as a basis to begin server capacity planning.

The list of specific hardware that the Windows 2000 DNS development and test team for server computers used during the previous testing included the following:

Hardware components	Sizing
Number of processors	Two
Processor	Intel Pentium II 400 MHz
Amount of RAM	256 MB (megabytes)
Hard disk drive space	4 GB (gigabytes)

These measurements were based on the server computer running a DNS server and with no other services in use. Where other hardware specifications or software configurations are used when deploying Windows 2000 DNS servers, your performance results are likely to vary from those documented here.

Server Capacity Planning

Planning and deploying DNS servers involves examining several aspects of the network and the capacity requirements for any DNS servers that will be used.

In many cases, adding more RAM to a DNS server can provide the most noticeable improvements in performance. This is because the DNS Server service fully loads all its configured zones into memory at startup. If the server is operating and loading a large number of zones, and dynamic updates are occurring frequently for zone clients, additional memory can be helpful.

The DNS server consumes memory as follows:

- Approximately 4 MB of RAM is used when the DNS server is started without any zones.
- For each addition of zones or resource records to the server, the DNS server consumes additional server memory. Its estimated that for the addition of every resource record to a server zone, an average of approximately 100 bytes of server memory is used.

For example, if a zone containing 1000 resource records is added to a server, it would consume approximately 100 KB (kilobytes) of server memory. he previous recommendations are not meant in any way to indicate maximum performance or limitations for Windows 2000 DNS servers.

Note: These numbers are approximate and can be influenced by the type of the resource records entered in zones, the number of resource records with the same owner name, and the number of zones in use at a specific DNS server.

Before a DNS namespace can be properly implemented in Windows 2000, the Active Directory services structure needs to be available. The recommended approach to the ADS and DNS design is to begin with the ADS design and then support it with the appropriate DNS namespace.

Active Directory design is an iterative process. It involves developing an initial ADS namespace and DNS architecture to support it, and then revising the ADS and DNS design if unforeseen, or undesirable consequences are uncovered.

The Windows 2000 Active Directory Namespace Design white paper describes the ADS namespace, including the forest and tree domain structure, organizational units, the global catalog, trust relationships, and replication. It then provides examples of namespace implementations and describes the architectural criteria that network architects and administrators should consider when designing an Active Directory namespace for the Enterprise. By following the recommendations in that paper, the Enterprise network architect should be able to design a namespace that is capable of withstanding company reorganizations without expensive restructuring.

Some of the fundamental DNS design questions that need to be answered are:

- How many Active Directory domains will you have?
- What will their names be?
- Will your DNS namespace have a private root?
- What will your computer names be?

Choosing Names

In Windows 2000, Active Directory domains are named with DNS names. When choosing DNS names to use for your Active Directory domains, identify the registered DNS domain name suffix that your company has reserved for use on the Internet, such as ‘company.com.’. It is recommended that you use different internal and external namespaces to simplify name resolution process. So, you could use internally (and as a forest root) a registered DNS suffix different from the external one, like “comp.com.”, or subdomain of the external domain, like “corp.company.com.”. You can then combine this name with a location or organizational name used within your company to form full names for your Active Directory domains, for example “hr.corp.company.com.”. This method of naming ensures that each Active Directory domain name is globally unique.

Once you have decided on DNS names for each of your Active Directory domains, you can use these names as parents for creating additional child domains to further manage other divisions within your company. Child domains must have DNS names that are immediately subordinate to their parent’s DNS name. For example, if a child domain were to be added in the ”us.corp.company.com.” tree for the human resources department in the American branch of the company, an appropriate name for that domain might be “hr.us.corp.company.com.”

Internet Access Considerations

Typically, a company namespace consists of two portions: private and public. The private one is a portion invisible from the outside world, while the public one is exposed to the Internet. Here the names that form the private and public namespaces are referred to as internal and external, respectively. Even though the private names are not exposed to the Internet, repetition of any external names (not only from the company, but from the Internet in general) in the private namespace is strongly discouraged, since it may lead to the ambiguity in name resolution processes.

In this section the focus is on the design of the private namespaces and the configuration of the DNS servers and zones. The specifics of two different designs are presented by considering two companies using private namespaces of different structure. These two companies, YYY and ZZZ Corporations, have reserved the DNS domain name suffixes, yyy.com. and zzz.com*.* The general approach to DNS configuration is to have internal (those that are accessible from internal clients only) and external DNS servers. External DNS servers contain the records that are supposed to be exposed to the Internet. The internal DNS namespace may contain a private root, in which case all internal clients that are anticipated to require name resolution must support Name Exclusion List or Proxy Autoconfiguration File to distinguish whether to direct name resolution queries to the proxy server or internal DNS server. An alternative approach is to configure internal DNS server(s) to forward to the Internet unresolved queries. Depending on the type of the clients that require DNS name resolution, the DNS configuration may be quite different. Four types of clients are distinguished based on their software proxy capability:

- proxy unaware,
- supporting LAT (Local Address Table),
- supporting Name Exclusion List, and
- Supporting Proxy AutoConfiguration file.

If name resolution is required by proxy unaware clients, or clients supporting only LAT, then the private DNS namespace can’t have a private root and one or more internal DNS servers must forward to the Internet unresolved queries.

As recommended in the previous section, the desired internal namespaces would be corp.yyy.com. and corp.zzz.com*.*

If the internal and external namespaces overlap, the configuration becomes more complicated. The example of such overlap is external web server www.yyy.com. and internal computer host1.yyy.com*.* This approach introduces some complications to the internal DNS configuration:

- to enable an internal computer to resolve the name of an external server and contact it, all clients must support Proxy AutoConfiguration File, unless external servers are cloned internally and external DNS records are copied internally (which increases the total cost of ownership due to required additional hardware and administration), or external DNS records are copied internally and the firewall is properly configured to enable internal clients to contact external servers,
- if all clients support Proxy AutoConfiguration File, then the file must be configured appropriately to distinguish internal and external computers with the same suffixes (as in the example above, with www.yyy.com. and internal computer host1.yyy.com.).

The following DNS configuration and name resolution scenarios are considered in detail with overlapping internal and external namespaces, since it is the most complicated case.

It is assumed that the namespaces of both companies consist only of names within a NSI assigned domain, that is, yyy.com. and zzz.com*.* It is also assumed that all computers in the YYY Corporation are proxy clients supporting Proxy AutoConfiguration File, while none of the computers in the ZZZ Corporation are proxy clients. The goal in this section is to demonstrate the appropriate configuration of the DNS servers, zones and clients to satisfy the following requirements:

- Expose only a public portion of the namespace to the Internet,
- Enable a company computer to resolve any (internal or external) names within its company,
- Enable a company computer to resolve any name from the Internet.

Finally, assume that the two considered corporations have merged and now every computer from these two private namespaces should be able to resolve any (internal and external) name, not only within the namespace of its own company, but within a namespace of the merged company as well.

The following solution will satisfy all four of these requirements.

Two DNS servers exposed to the Internet are authoritative for two zones, yyy.com. and zzz.com., as shown on the figure below. (To simplify the example, one server and one zone per company have been chosen. In reality a company may choose to have more servers and zones such as first.yyy.com, second.yyy.com. and so forth.) These zones contain only records corresponding to external names and delegations of the YYY and ZZZ Corporations (or in other words, only those records which these two companies wish to expose to the external world). This is the only common solution for both companies. The rest of the design features are different.

First consider the private namespace design and the configuration of the DNS servers, zones and clients in case the company’s computers are not proxy clients, for example, in ZZZ Corporation.

A company must devote a set of DNS Servers that are not exposed to the Internet to maintain zones containing all names (both internal and external) from the company namespace. Every DNS client must send DNS queries to some of these DNS servers. Every DNS server must forward queries to a pre-assigned forwarder (s). If a DNS server contains a top-level company namespace zone, that is, zzz.com., then its forwarder is a DNS server(s) exposed to the Internet. The communication between internal and external servers takes place through a firewall. Every other internal DNS server forwards unresolved queries to a DNS server(s) that contains the top-level company namespace zone.

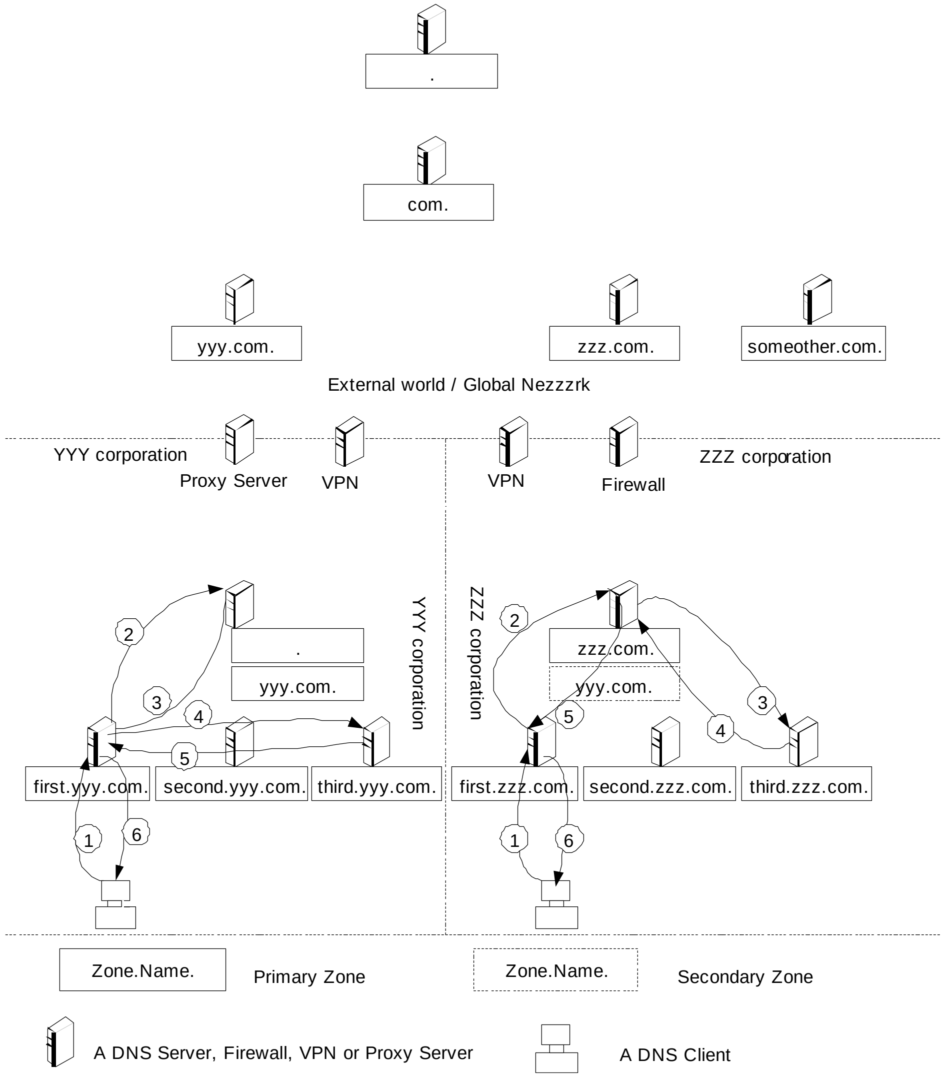
To guarantee that a company client is able to resolve any hostname from the merged companies every DNS server containing a top-level company namespace zone, that is, zzz.com., must also contain the zones containing all (internal and external) names of the merged companies.

Now take a look at a private namespace design and the configuration of the DNS servers, zones and clients for the YYY Corporation. The private namespace includes a private root,”.”.

A company must devote a set of DNS servers that are not exposed to the Internet to maintain zones containing internal names from the private company namespace. Every DNS client submits a query to some (preferred or alternative) DNS servers or to the proxy server(s) based on the Proxy AutoConfiguration File (PAC File). Every internal DNS server contains in its Root Hints the address(es) of the private root DNS server(s).

To guarantee that a company client is able to resolve any hostname from the merged companies the “.” zone must contain delegations to the top-level zones of the merged companies private namespaces.

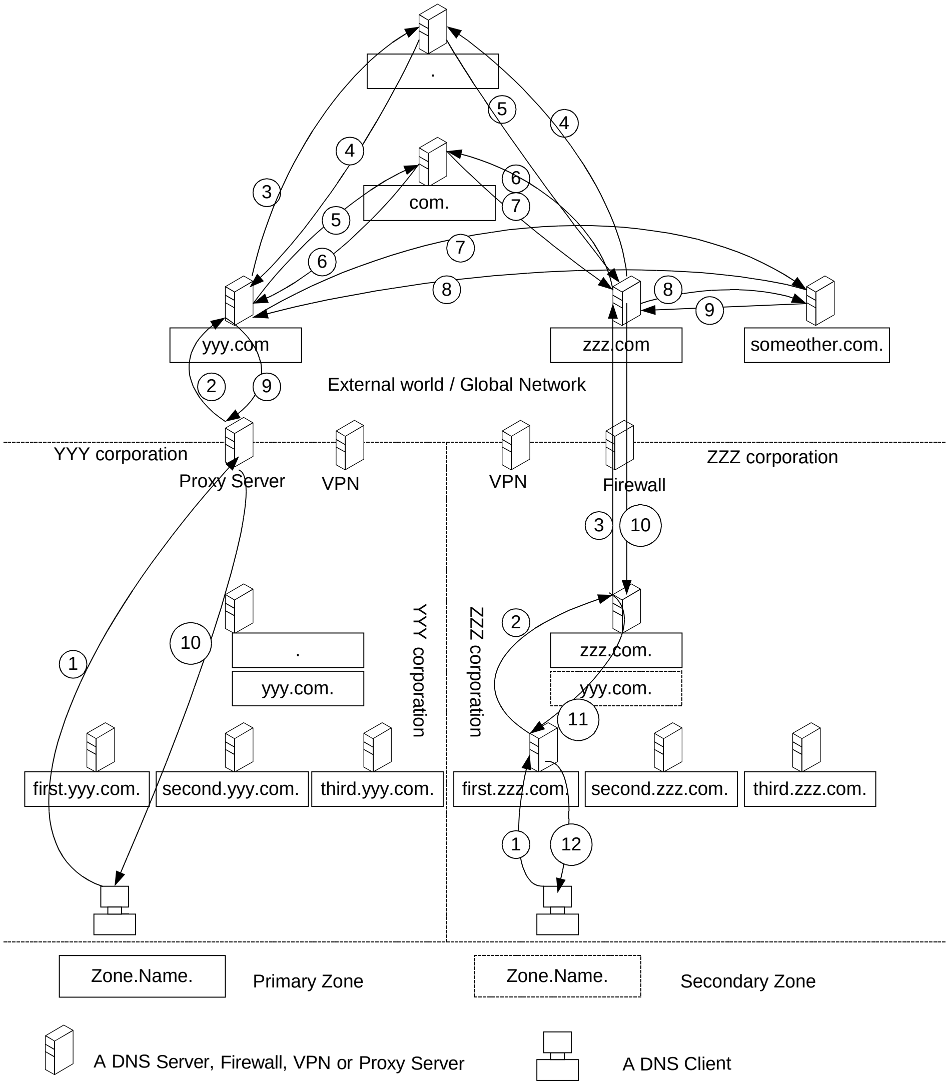
The following examples of queries demonstrate the internal and external names in both corporations, including satisfaction of all requirements listed above.



Starting with an example when a corporate computer needs to resolve an internal name (follow the above figure for illustrations).

A computer in the YYY Corporation needs to resolve a DNS query for www.third.yyy.com. First it finds that the name www.third.yyy.com. is internal based on PAC file. Therefore, it submits the query to the assigned DNS server (Step 1). If this DNS server is authoritative for the name www.third.yyy.com. or the cache contains necessary data, then the server will respond to the client. Otherwise the server will query a root server (Step 2). A root server returns a reference to the authoritative server (Step 3). Then the server sends a query to the authoritative server zone (Step 4), receives a response from it (Step 5) and finally passes it to the client (Step 6).

A computer in the ZZZ Corporation needs to resolve a DNS query for www.third.zzz.com. It submits the query to the assigned DNS server (Step 1). If this DNS server is authoritative for the name www.third.yyy.com. or the cache contains necessary data, then the server will respond to the client. Otherwise the server forwards the query to the DNS server containing the zzz.com. zone (Step 2). This server finds a delegation to the third.zzz.com. in the zzz.com. zone. It sends the query to that server (Step3) receives back the response (Step 4), passes it to the previous server (Step 5), which finally returns it to the client (Step 6).

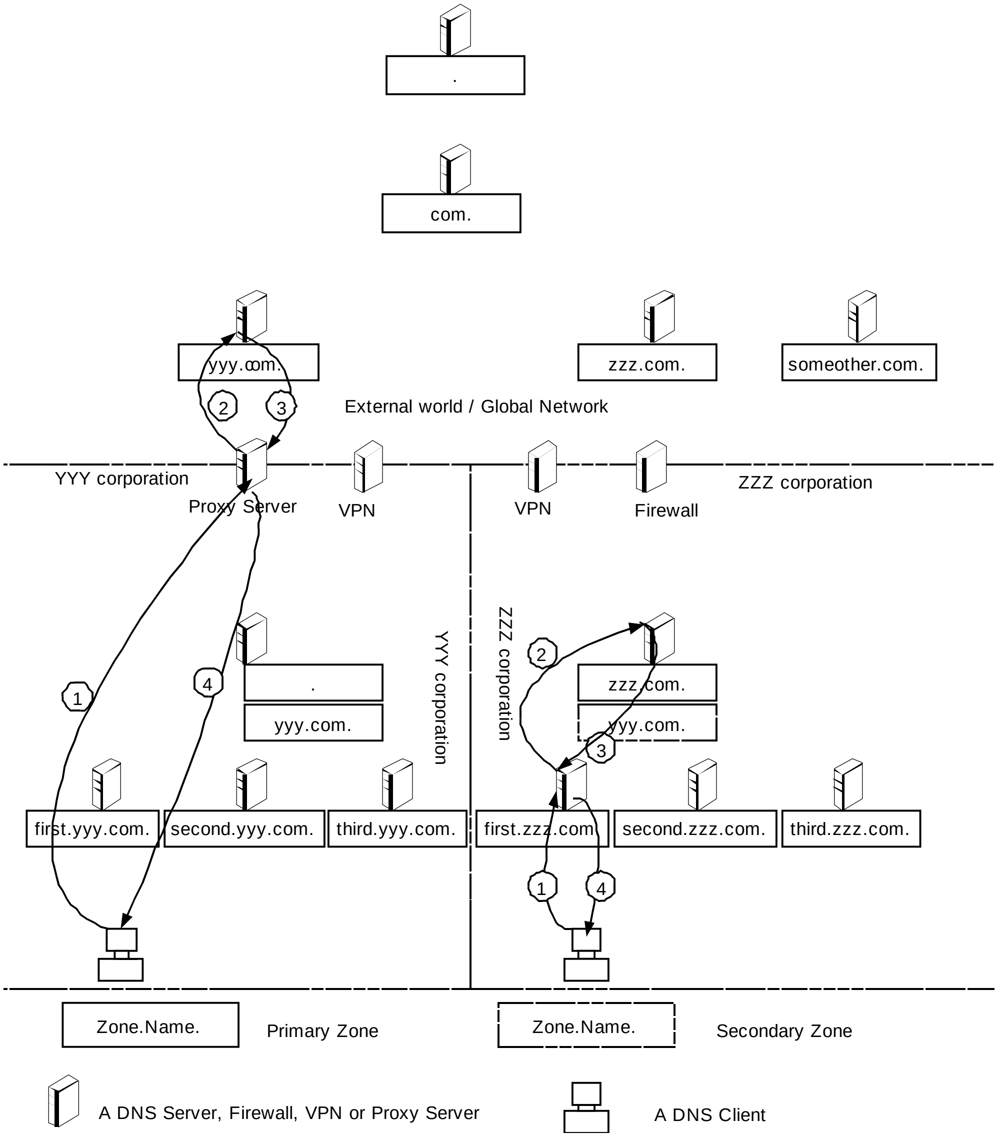


Now consider the example of a corporate computer that needs to resolve an external name (that does not belong to its company).

A computer in the YYY Corporation needs to open a web page on the www.someother.com. machine. Since it is a proxy client it sends a request to the proxy server (Step 1) after it finds that the name www.someother.com. is external based on the PAC file. The proxy server sends a DNS query to the assigned DNS server (Step 2) which recursively resolves the query. It sends a query to the root server (Step 3) and receives a reference to the server that contains the com. zone (Step 4). Then it sends the query to that server (Step 5) and receives a reference to the server that contains a zone someother.com. (Step 6). It sends a query to the latter (Step 7), which resolves the query and returns the response to the server

(Step 8). The DNS server returns the response to the proxy server (Step 9). Finally, the proxy server uses the obtained IP address of www.someother.com. to contact it and provides the necessary information to the client (Step 10).

A computer in the ZZZ Corporation needs to resolve a DNS query for www.someother.com*.* It submits an appropriate query to the assigned DNS server (Step 1). If its cache contains necessary data, then the server will respond to the client. Otherwise the server forwards the query to the DNS server containing the zzz.com. zone (Step 2). This server forwards the query to the external server (Step 3) through the firewall. The latter performs the name resolution similar to the previous case (Steps 4-9) and passes the result to the client though the firewall and the chain of participated in the resolution servers (Steps 10-12). Then the client uses the resolved IP address of www.someother.com. to contact it through the firewall and download the desired web page.

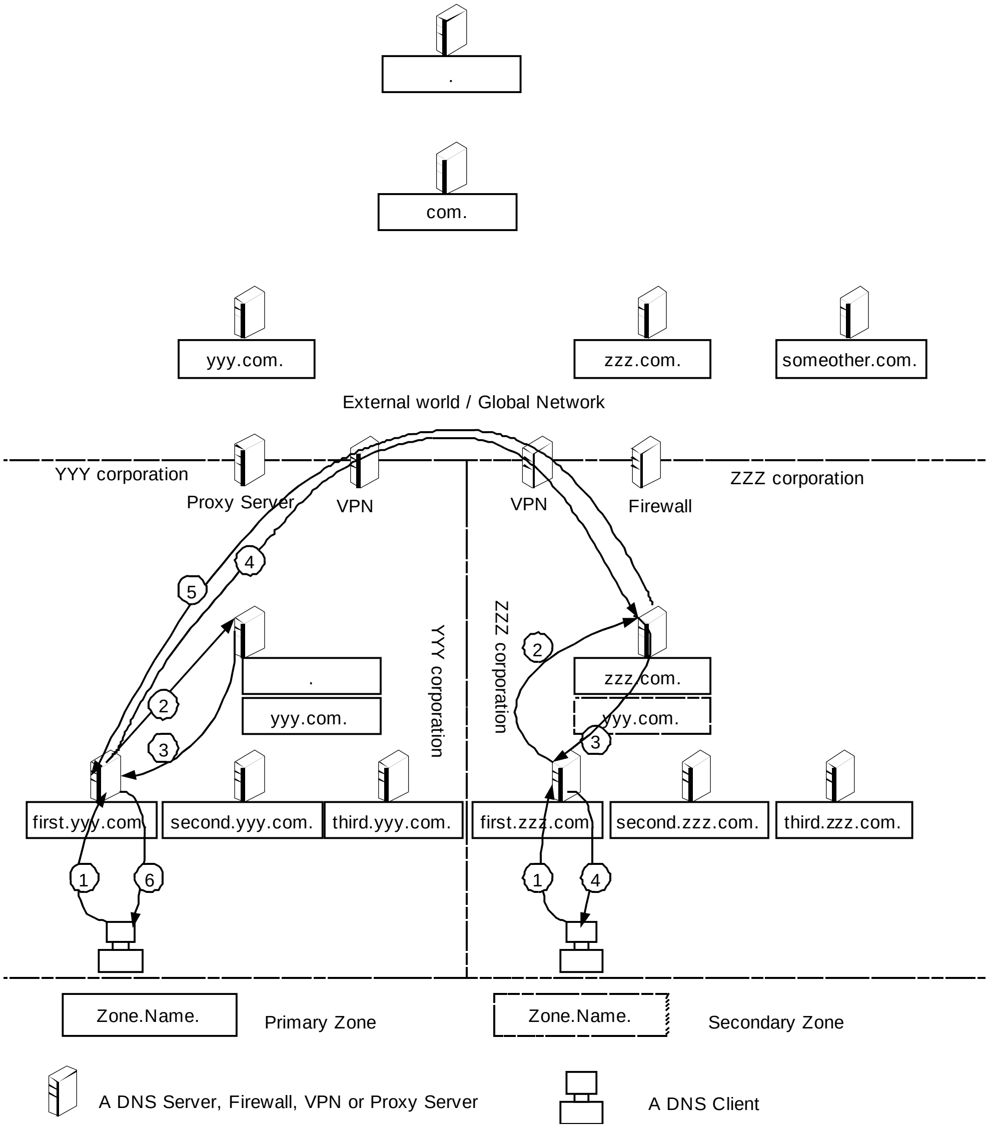


Now consider an interesting case of a corporate computer that needs to resolve an external name of a computer from its own company.

A computer in the YYY Corporation needs to open a web page on the www.yyy.com. machine. Since it is a proxy client it sends a request to the proxy server (Step 1) after it finds that the name www.yyy.com. is external, based on the PAC file. The proxy server sends a DNS query to the assigned DNS server (Step 2) which happens to be authoritative for www.yyy.com. The DNS server resolves the query and returns the response to the proxy client (Step 3). Finally the proxy server uses the obtained IP address of www.yyy.com. to contact it and provides necessary info to the client (Step 4).

A computer in the ZZZ Corporation needs to resolve a DNS query for www.zzz.com. It submits the query to the assigned DNS server (Step 1). If its cache contains the necessary data, the server will respond to the client. Otherwise the server forwards the query to the DNS server containing the zzz.com. zone (Step 2). Since the server is authoritative for the name www.zzz.com. it resolves the query and returns the response to the client through the forwarding DNS server (Steps 3-4). At this point it is important to emphasize the significance of the fact that the zzz.com. zone on the internal DNS server contains both internal and external names. If it contained only internal names then this query would not be resolved and name error would be returned to the client.

Finally, consider an example of a corporate computer that needs to resolve a host name from the private namespace of a merged company.



A computer in the YYY Corporation needs to contact a computer myname.zzz.com.

First it finds that the name myname.zzz.com. is internal, based on the PAC file. Therefore, it submits a query to the assigned DNS server (Step 1). If the cache contains the necessary data, the server will respond to the client. Otherwise, the server will query a root server (Step 2). The root server that contains the "." zone finds a delegation to the zzz.com. zone and returns a reference to the authoritative server (Step 3). The server uses the IP address of the name server that contains the zzz.com. zone to submit the query (Step 4). Since that server is authoritative for myname.zzz.com., it resolves the query and returns the answer (Step 5). Finally, the server returns response to the client (Step 6).

A computer in the ZZZ Corporation needs to resolve a DNS query for myname.yyy.com*.* It submits a query to the assigned DNS server (Step 1). If its cache contains the necessary data, the server responds to the client. Otherwise, the server forwards the query to the DNS server containing the zzz.com. zone (Step 2). Since this server contains a secondary copy of the zone yyy.com. it resolves the query and returns it to the client through the previous server (Steps 3-4).

Each of the two suggested solutions has disadvantages associated with it.

The solution of company YYY requires maintenance of the PAC file.

At the same time, the solution of company ZZZ puts a significant load on the internal DNS servers containing top-level private namespace zones. This is because the majority of the queries generated within the company are forwarded to these servers. Moreover, in the case of the same internal and external namespaces, these servers contain larger zones, since they must contain both internal and external names.

Characters in Names

As mentioned above, the standard characters for DNS, according to RFC 1123, are A-Z, a-z, 0-9 and the -. In organizations that have an extensive investment in Microsoft NetBIOS technology, the names conform to the NetBIOS standard. These organizations should seriously consider moving towards DNS standard.

The process of adjusting your naming conventions may prove to be time consuming. In an attempt to ease migration from Windows NT 4.0 NetBIOS names to Windows 2000 DNS names, Windows 2000 DNS includes support for extended ASCII and Unicode characters. However, the support for additional characters can only be taken advantage of in a pure Windows 2000-based network environment, since most third party resolver software, such as Unix or Apple is RFC 1123 standards-based.

Note: If a non-standard DNS name is entered during Windows 2000 DNS setup, the warning message will appear suggesting the standard DNS name.

Computer Names

Windows NT 4.0 and previous versions of the operating system use a NetBIOS name to identify a particular machine on the network. A Windows 2000-based machine can be identified by a NetBIOS name (for down-level interoperability), and a full DNS computer name, which is a concatenation of Host name and primary DNS suffix. The primary DNS suffix is part of the base machine configuration and is not related to any networking components. Non-networked or non-TCP/IP-based machines do not have primary DNS suffix. By default the primary DNS suffix of a computer is set to the DNS domain name of the Active Directory to which it is joined. To change the primary DNS suffix of a computer, a computer administrator should click **System** in Control Panel, click the **Network Identification** tab, click **Properties**, click **More**, and then enter a suffix in the Primary DNS suffix of this computer box. Primary DNS suffix could be also assigned to a group of computers through the group policy.

The table below contains comparison between a NetBIOS name and a DNS Hostname.

	NetBIOS name	**Full computer name**
Type	Flat	Hierarchical
Character Restrictions	A-Z, a-z, 0-9, whitespace, Unicode chars, symbols: ! @ # $ % ^ & ') (. - _ { } ~	A-Z, a-z, 0-9, symbols: -_, Unicode chars. The dot, '.', has label separator meaning
Maximum Length	16 bytes (including one reserved byte)	63 UTF-8 bytes per label 255 UTF-8 bytes for whole name
Name Service	NBNS (WINS and broadcast)	DNS

Thus, the NetBIOS name is restricted to 15 bytes, whereas a Host name can be up to 63 bytes long (DNS names are encoded in UTF-8 and are not necessarily one byte per character).

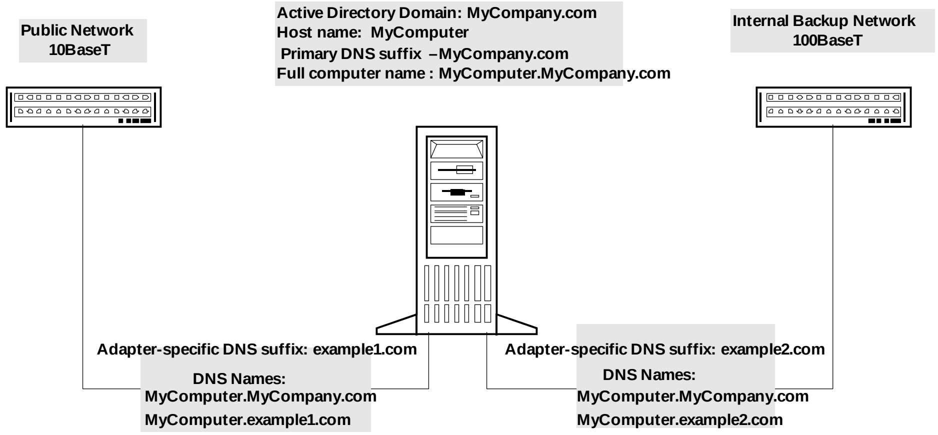
The Network Identification property page contains the following entries:

- Full Computer Name: MyComputer.MyCompany.com.
- Member of Domain: MyCompany.com.

In this example, the “MyComputer” is the Host name and NetBIOS name, while “MyCompany.com” is the primary DNS suffix.

Per-Adapter Naming

A machine with multiple adapters can acquire different domain names as part of the adapters’ IP configuration. The adapters of the machine can then be addressed on an individual basis by their Hostnames. An example of this configuration is shown below.


Public Network
10BaseT
Active Directory Domain: MyCompany.com
Host name: MyComputer
Primary DNS suffix –MyCompany.com
Full computer name : MyComputer.MyCompany.com
Internal Backup Network
100BaseT
Adapter-specific DNS suffix: example1.com
DNS Names:
MyComputer.MyCompany.com
MyComputer.example1.com
Adapter-specific DNS suffix: example2.com
DNS Names:
MyComputer.MyCompany.com
MyComputer.example2.com

In the picture above, a machine with the MyComputer Host name is joined to the *MyCompany.com.* AD domain. Its primary DNS suffix is also set by default to *MyCompany.com.*

The first adapter, which is being used for public access, is configured with the *example1.com.* DNS suffix. The second adapter, which is used exclusively for backups, has the *example2.com.* DNS suffix. The machine, therefore, can be accessed publicly through the first adapter using the *MyComputer.example1.com.* DNS name. For backup purposes the same machine can be accessed through the second adapter using the *MyComputer.example2.com.* DNS name.

Integrating ADS with Existing DNS Structure

In order for a DNS server to be able to support the Active Directory it is required to support the SRV records and it is recommended to support the dynamic updates, as described in the RFC 2136.

When integrating ADS into an existing DNS infrastructure, the decision needs to be made whether the Active Directory namespace will join, or overlap the existing DNS namespace.

If there is no overlap, you can delegate a new Windows 2000 DNS namespace from the existing DNS structure. When a DNS namespace is delegated off an existing DNS tree, the DNS server that owns the zone file for the newly delegated namespace, and becomes the primary master for that namespace. The DNS zone name, that has been delegated, should correspond to the ADS root domain. This approach is not required, but recommended if you want to use the benefits of the Windows 2000 DNS server. You may continue using the existing DNS server without delegating the Active Directory namespace as long as current DNS servers support the SRV records and the dynamic updates.

If the overlap is inevitable, then the approach you should take depends on whether the existing DNS tree is implemented using Windows NT 4.0 DNS, or a non-Microsoft product.

If existing DNS tree is implemented by Windows NT 4.0 DNS, the solution is to upgrade the Windows NT 4.0 DNS servers to the Windows 2000 implementation of DNS.

If a non-Microsoft DNS implementation is in place and it does not support SRV RRs and Dynamic Update, then the question is: can it be upgraded. **Note**: The Dynamic Update feature is not required, but strongly recommended.

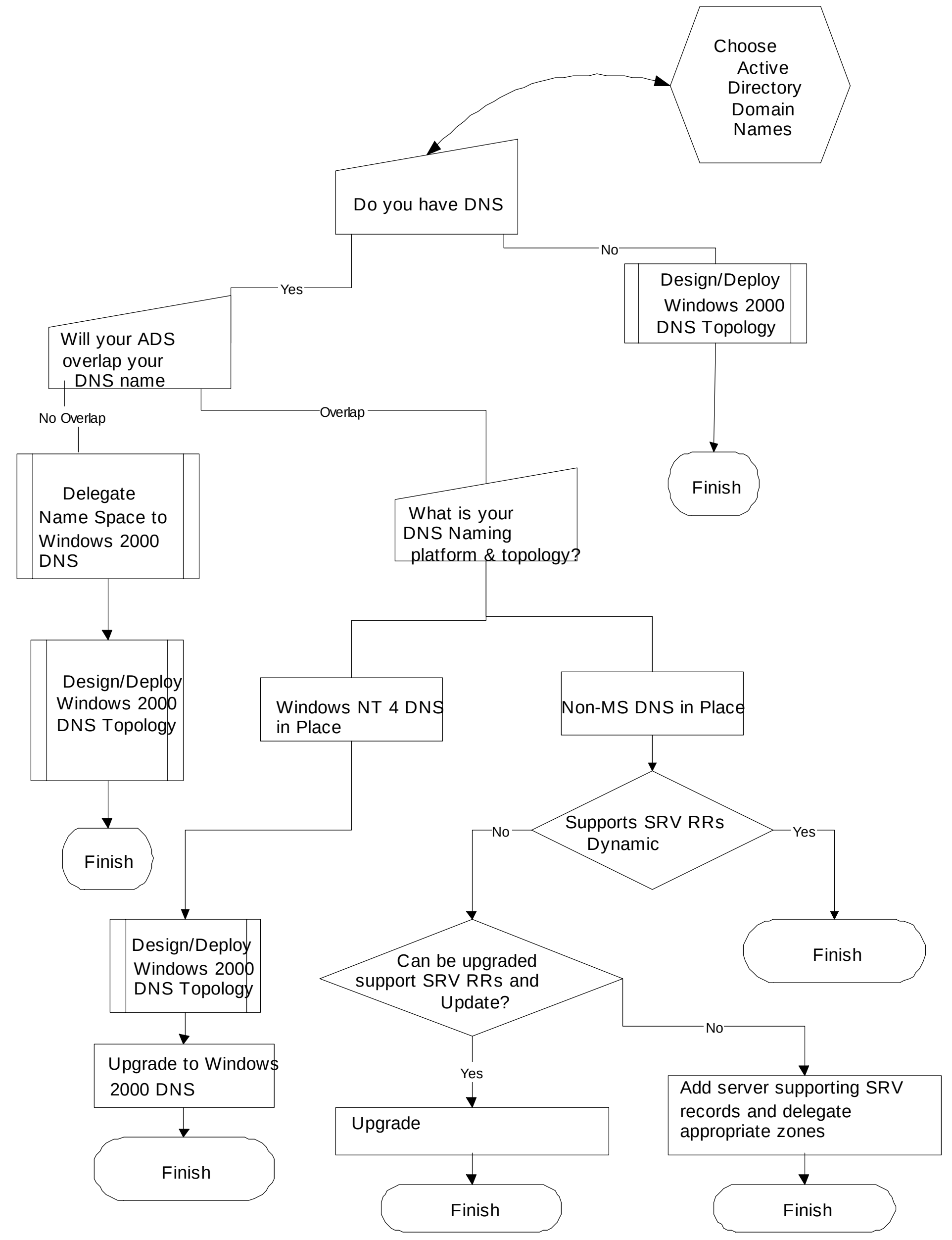
If existing non-Microsoft DNS servers can be upgraded, then perform the upgrade.

If existing non-Microsoft DNS servers cannot be upgraded, add another DNS server that does support SRV records and dynamic updates and delegate certain zones to this server.

On the DNS server that does not support SRV records and dynamic update, delegate the following zones to the DNS server that does: `_tcp.`*`<Active Directory domain name>`*, `_udp.<Active Directory domain name>`, `_msdcs.<Active Directory domain name>` and `_sites. <Active Directory domain name>.`

On the DNS server that does support these features, create and then enable dynamic update on each of the zones in the preceding list. Active Directory dynamically updates the appropriate records in these zones.

The process of integrating ADS into an existing DNS infrastructure is better understood from the following flowchart:


Choose Active Directory Domain Names
Do you have DNS
No
Yes
Design/Deploy Windows 2000 DNS Topology
Finish
Will your ADS overlap your DNS name
No Overlap
Overlap
Delegate Name Space to Windows 2000 DNS
What is your DNS Naming platform & topology?
Design/Deploy Windows 2000 DNS Topology
Finish
Windows NT 4 DNS in Place
Non-MS DNS in Place
Supports SRV RRs Dynamic
No
Yes
Finish
Design/Deploy Windows 2000 DNS Topology
Can be upgraded support SRV RRs and Update?
No
Yes
Upgrade to Windows 2000 DNS
Finish
Upgrade
Add server supporting SRV records and delegate appropriate zones
Finish
Finish

Migration to Windows 2000 DNS

The first step in migrating non-Microsoft DNS servers to the Windows 2000 implementation of DNS is to introduce Windows 2000 DNS servers as secondary servers for the overlapping zones. One of the key points here is to configure a zone transfer from a master to a secondary Windows 2000 DNS server and make sure that the zone transfer process does not generate any errors. Errors can occur if during the zone transfer the Windows 2000 DNS server is not able to recognize records sent by the non-Microsoft DNS server. These records should either be repaired or removed from the zone in order for the zone transfer to complete successfully.

Once the Windows 2000 DNS servers have stabilized in the new role, their

SUMMARY

secondary zones can be upgraded to DS integrated zones. At this point non-Microsoft DNS servers can be safely retired and removed from the network.

Deploying DNS to Support Active Directory

If you are designing a brand new network environment, the process of deploying Active Directory service/Windows 2000 DNS is relatively straightforward. Chances are, however, that the Active Directory service you are designing will need to be integrated into existing DNS infrastructure.

Partitioning, and Replication (Choosing your Zones)

When designing a DNS namespace for an Active Directory, the emphasis should be placed on creating an effective partition and replication topology while keeping replication and update traffic at bay.

The following domain/zone configuration is recommended:

- Each Active Directory domain should have a DNS zone corresponding to the name of the domain. This zone should be configured on a DNS server running on the Domain Controllers in that Active Directory domain. The zone should be Active Directory-integrated.
- DNS servers should running on at least two domain controllers in each Active Directory domain and at least one Domain Controller in each site.
- Since most of the records ending with `“_msdcs.<DnsForestName>”` suffix should be accessible through entire forest it could be useful to delegate a zone `“_msdcs.<DnsForestName>”` from the zone `“<DnsForestName>”`. All DNS servers in the enterprise that are connected to the primary for `“_msdcs.<DnsForestName>”` zone servers, over slow or not-permanent links, should be configured as secondary servers for the `“_msdcs.<DnsForestName>”` zone. One DNS server from each site should be configured to poll `“_msdcs.<DnsForestName>”` zone transfer from a primary server. All other DNS server in a site poll the zone transfer from the chosen DNS server in that site. The primaries should not notify secondaries of any changes in the zone. The secondaries will pool updates from the primaries at zone refresh intervals. The DNS server that polls the zone transfer directly from the primary server should be configured to notify all other DNS servers in the same site. This configuration doesn’t flood the network with the zone replication traffic while enabling clients in the child domains to resolve DNS queries addressed to the `“_msdcs.<DnsForestName>”` zone when the link is down.

The configuration of the reverse lookup zones is not based on the Windows 2000 Domain structure. Instead it is based on the range of IP addresses assigned to a company. If a company is assigned B class IP addresses such as 172.56.X.Y. then a reverse lookup zone of 56.172.in-addr.arpa. will be created. It may contain delegations to other domains, such as, 1.56.172.in-addr.arpa., 2.56.172.in-addr.arpa. and so on. It is also possible to configure classless reverse lookup zones that as described in the Internet Draft “Classless IN_ADDR.ARPA delegation”.

Using Automatic Configuration

The Windows 2000 implementation of DNS offers a DNS Server Configuration wizard, which greatly simplifies the DNS server installation and configuration process. For example, it offers an elegant way of priming the root hints for a new DNS server.

The Server Configuration Wizard sends to the computer's preferred and (possibly alternative) DNS server(s) a NS query for the root, ".", node. The response is placed into the root hints of this new server. If no root servers are detected, then the wizard sends the same query to the DNS servers specified in the cache.dns file, corresponding to the root servers on the Internet. If again no root servers are detected, the wizard prompts the user to either make the server a root server (by simply choosing the appropriate option) or manually specify root hints.

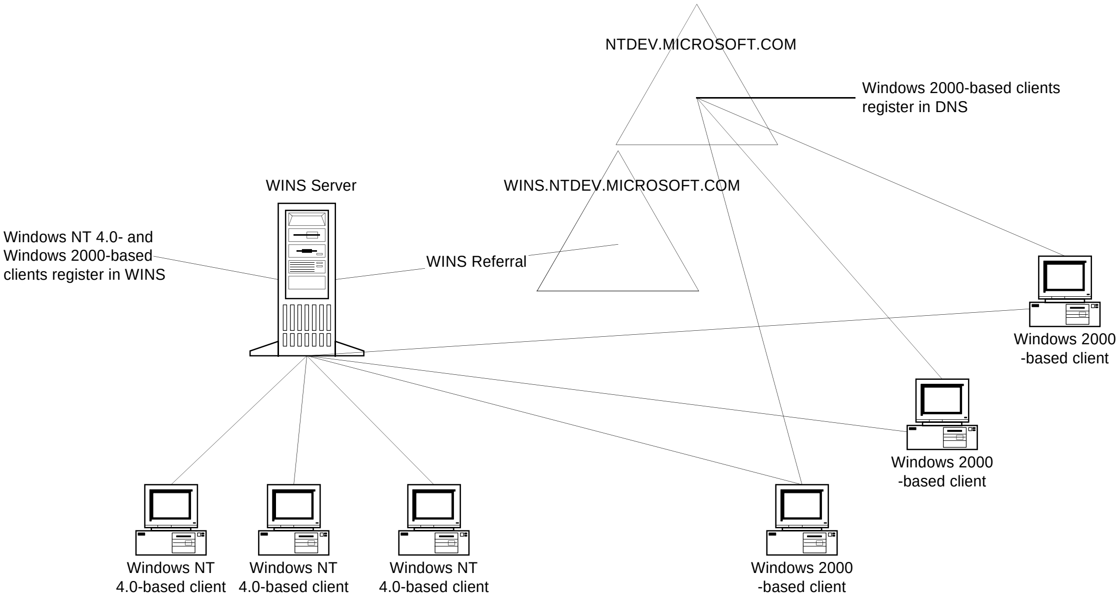
WINS Referral

WINS filled the role of domain and machine locator service for previous versions of Windows NT. Windows 2000 will not require WINS in a NetBIOS-less environment. However, WINS will always be required in a mixed environment where Windows 2000-based machines interoperate with other systems such as Windows NT 4.0, Windows 9X, and Windows for Workgroups.

WINS Referral is the recommended way for Windows 2000 DNS clients to address down-level machines registered in WINS. Because Windows 2000 resolvers are optimized to use DNS, they would be much more efficient looking up down-level clients in a DNS database as opposed to WINS database. To enable this kind of lookup, a WINS referral zone can be created in DNS that points to the WINS database.

This zone does not perform any registrations or updates; it simply refers DNS lookups to WINS.

Whenever Windows 2000-based clients send a query with the unqualified name (for example, *ntservermydomain*), the default domain name suffix will be tried first. Additional suffixes, however, can be supplied as part of the DHCP configuration. If the name of the WINS Referral zone is one of them, all WINS client names will be able to be resolved.


NTDEV.MICROSOFT.COM
Windows 2000-based clients register in DNS
WINS Server
WINS.NTDEV.MICROSOFT.COM
Windows NT 4.0- and Windows 2000-based clients register in WINS
WINS Referral
Windows 2000 -based client
Windows 2000 -based client
Windows NT 4.0-based client
Windows NT 4.0-based client
Windows NT 4.0-based client
Windows 2000 -based client

In the picture above, a WINS referral zone called *wins.mydomain.microsoft.com.* has been created and pointed to the WINS database. Assume that a Windows NT 4.0-based client has a name *client1*. A Windows 2000-based client belongs to the *mydomain.microsoft.com.* If the Windows 2000-based client has received a *wins.mydomain.microsoft.com.* suffix with its DHCP configuration, then in an attempt to resolve an unqualified name client1, it will first try the *mydomain.microsoft.com.* suffix (that is, client1.mydomain.microsoft.com.), and if that fails, it will then try *wins.mydomain.microsoft.com.* (that is, client1.wins.mydomain.microsoft.com.). When the DNS server authoritative for the *wins.mydomain.microsoft.com*. zone receives the query it can't resolve the requested name. But since it is configured to use WINS look-up it submits a query for client1 to the WINS server. The WINS server containing appropriate registration returns the host IP address to the DNS server and that passes it to the Windows 2000-based client.

Microsoft has chosen DNS to be its strategic name space in Windows 2000 replacing NetBIOS used as a name service in previous versions of Windows NT.

The implementation of DNS in Windows 2000 is a unique DNS Server implementation that is fully interoperable with other standards-based implementations of DNS Server. It is a scalable, highly available, and high performance solution. The following features of Windows 2000 DNS make it a good choice for the corporations looking to implement a reliable hierarchical distributed network environment:

- ADS Integration
- IXFR
- Dynamic Update and Secure Dynamic Update
- Unicode Character Support
- Enhanced Domain Locator

- Enhanced Caching Resolver Service
- Enhanced DNS Manager

To properly deploy DNS in the Windows 2000-based environment, it is recommended to start with the ADS design and then support it with the appropriated DNS namespace. For ADS design refer to the Windows 2000 Active Directory Namespace Design white paper.

For More Information

For the latest information on Windows 2000 visit our World Wide Web site at http://www.microsoft.com/windows2000

AXFR–Type of zone file replication. AXFR replicates the entire zone. (See also IXFR.)

Authoritative DNS server–A DNS server is considered authoritative for a name if it loads the zone authoritative for that name.

Authoritative DNS zone–A DNS zone is considered authoritative for a name if the name belongs to the DNS sub-tree, delegated to that zone.

DNS–Domain Name System.

IXFR–Type of the zone file replication. IXFR, incremental zone transfer, replicates only the changed records of the zone file

Master and Slave DNS servers–Two DNS servers are called Master and Slave if they contain the copies of the same zone, one of which is directly replicated from another. The source of replication is called Master server, the destination of replication is called Slave server. Every Master may have one or more Slaves and vice versa, every Slave may have one or more Masters. The same DNS server may be the Master and Slave at the same time.

Primary and Secondary zones–The same zone may be represented by primary and secondary copies. The primary is the zone/copy that allows direct updates of its resource records. The secondary is the one, that receives all the updates from primary or secondary zones through the zone transfer mechanism only. Only the DS integrated zones may have multiple primaries. Multiple secondaries are allowed in either scenario.

Resource Record–Atomic unit of the DNS database. All resource records have the same format that includes NAME, TYPE, CLASS, TTL, RDLENGTH and RDATA that depends on TYPE and CLASS of the resource record. A set of resource records builds up a DNS zone.

Root Server–A DNS server that contains a root zone is called a root server.

Root Zone–A zone that contains the DNS root domain is called the root zone.

TTL–Time-To-Live (TTL) is a duration of time when a specific resource record could be cached.

UCS-2–Also known as Unicode is a character encoding protocol.

UTF-8–A character encoding protocol, specified in RFC 2044

WINS–Windows Name System (WINS) is the pre-DNS name system. It is still supported in the Windows 2000 in order to maintain interoperability between the different generations of Windows computers.

Zone Transfer–Process of replication of the zone from Master to Slave server.